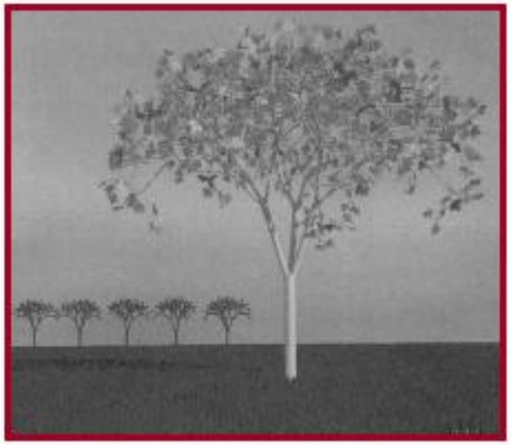
Procedural Modeling

- Various techniques and technical details

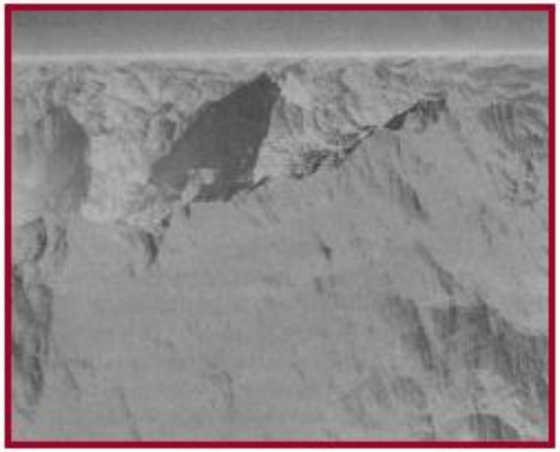
Modeling

- How do we ...
  - Represent 3D objects in a computer?
  - Construct such representations quickly and/or automatically with a computer?
  - Manipulate 3D objects with a computer?



H&B Figure 10.79

Fowler



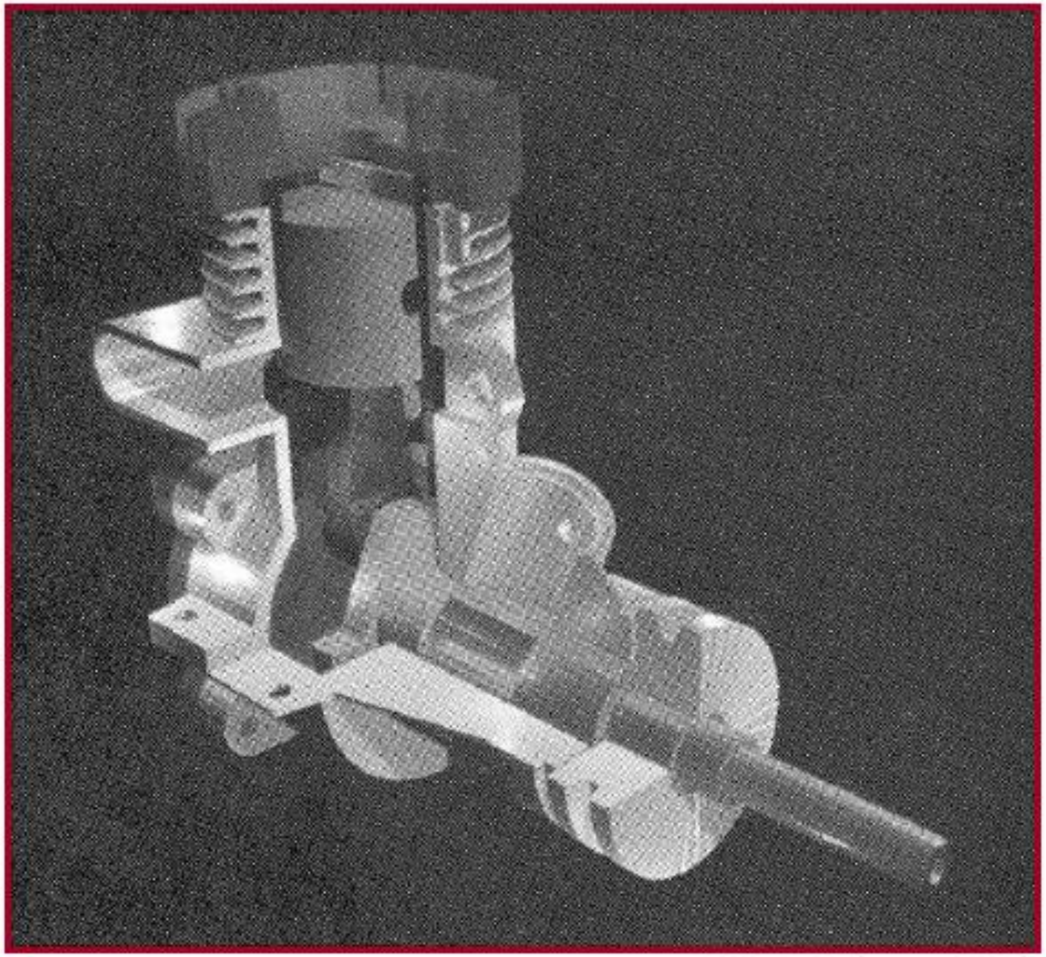
H&B Figure 10.83b

Model Construction

- Interactive modeling tools
  - CAD programs
  - Subdivision surface editors
- Scanning tools
  - CAT, MRI, laser, magnetic, robotic arm, etc.
- Computer vision
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Interactive Modeling Tools

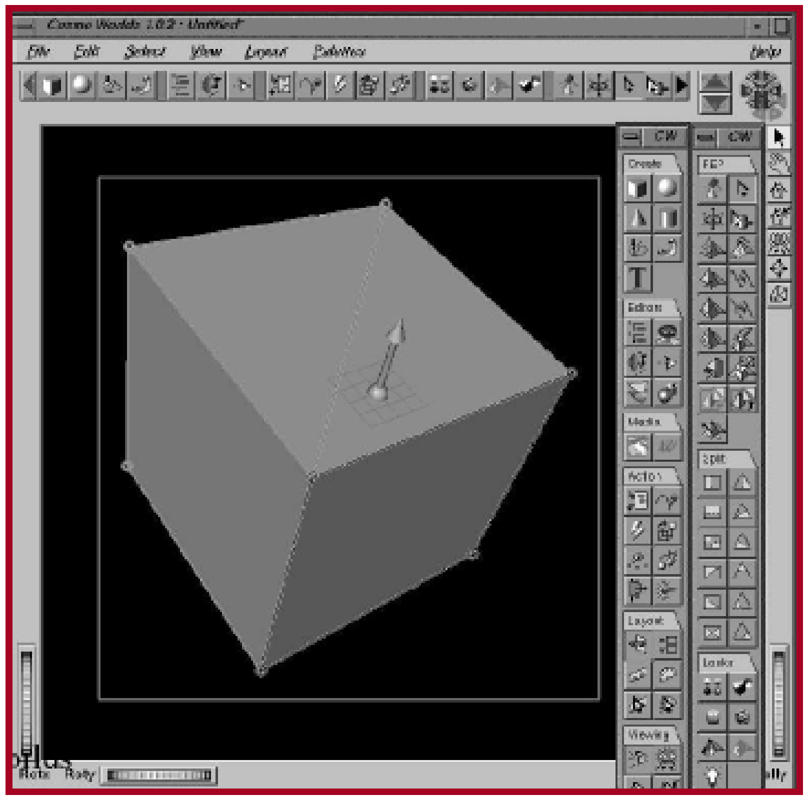
- Example: Mechanical CAD



H&B Figure 9.9

Interactive Modeling Tools

- User constructs objects with drawing program
  - Menu commands, direct manipulation, etc.
  - CSG, parametric surfaces, quadrics, etc.



Cosmoworlds, SGI

Procedural Modeling

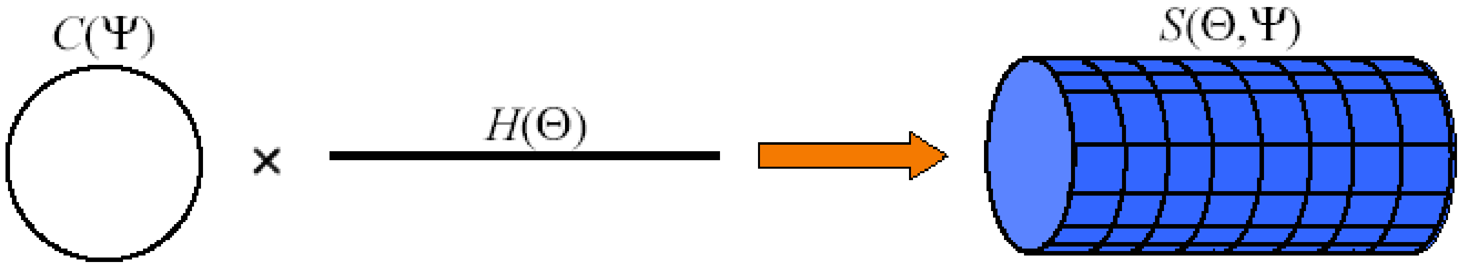
- Goal:
  - Describe 3D models algorithmically
- Best for models resulting from ...
  - Repeating processes
  - Self-similar processes
  - Random processes
- Advantages:
  - Automatic generation
  - Concise representation
  - Parameterized classes of models

Procedural Modeling

- Sweeps
- Fractals
- Grammars

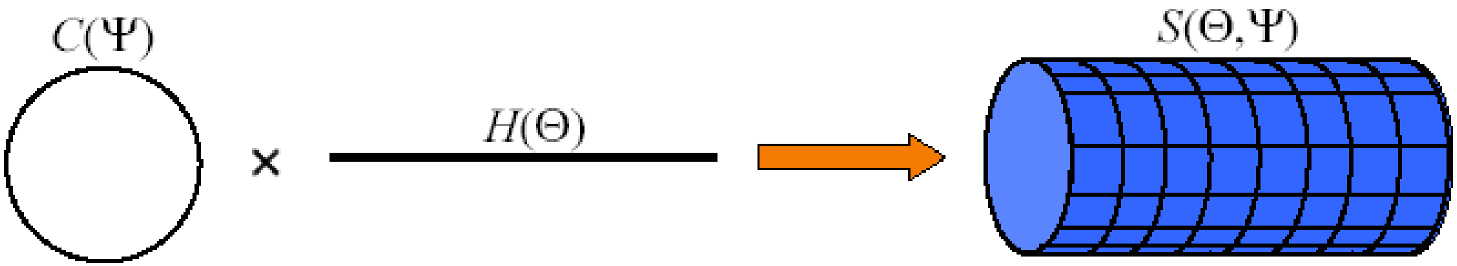
Sweeps

Given a 3D sweep curve $H(\Theta)$ and a 2D generating curve $C(\Psi)$, we define the sweep surface $S(\Theta,\Psi)$ as the sweep of C along H:



Sweeps

Given a 3D sweep curve $H(\Theta)$ and a 2D generating curve $C(\Psi)$, we define the sweep surface $S(\Theta,\Psi)$ as the sweep of C along H:



In this example, the sweep curve is simply used to translate the generating curve:

$$S(\Theta,\Psi) = H(\Theta) + C(\Psi)$$

We can define more complex sweep surfaces.

Example: Seashells

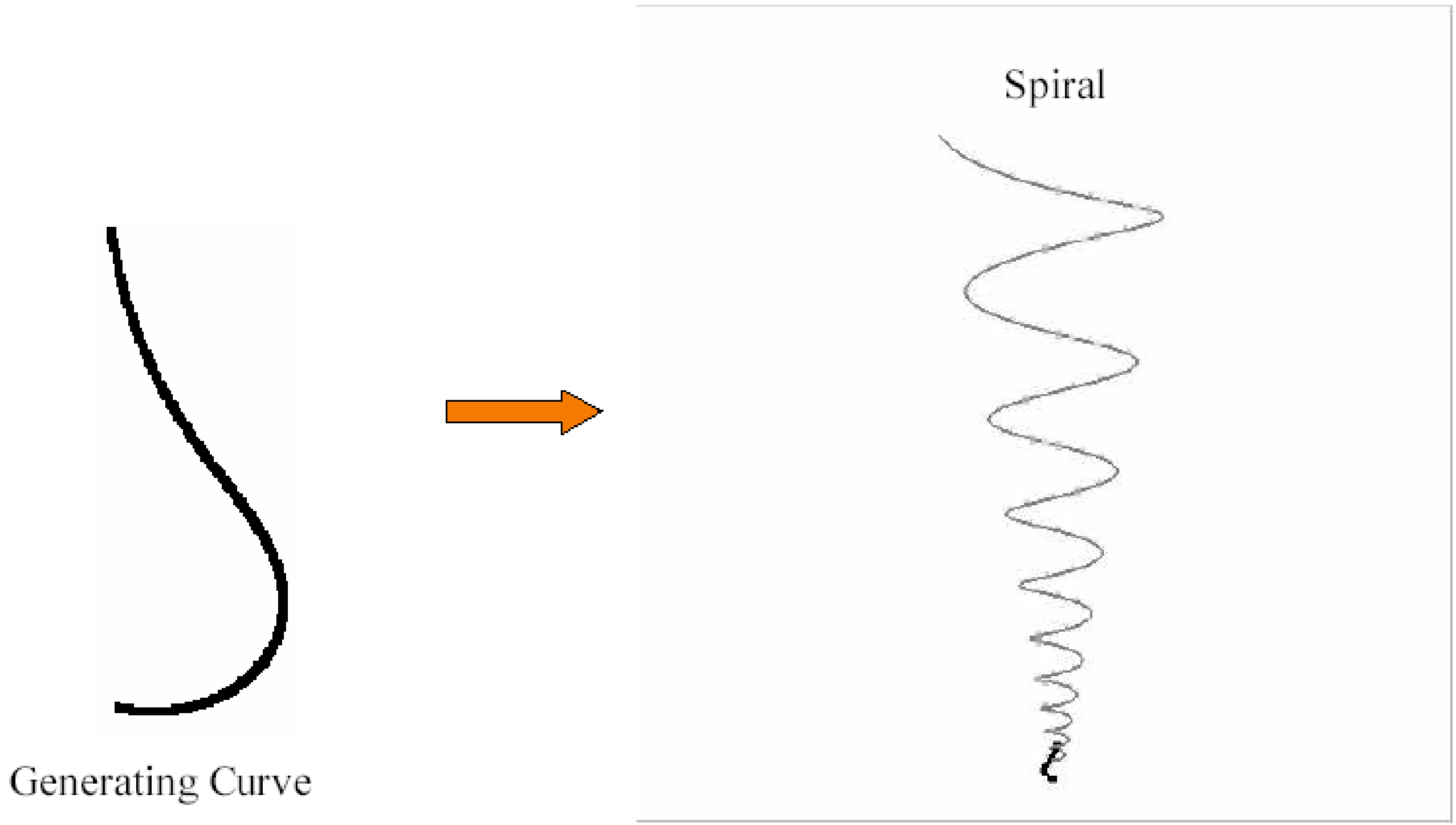
- Create 3D polygonal surface models of seashells

"Modeling Seashells,"
Deborah Fowler, Hans Meinhardt,
and Przemyslaw Prusinkiewicz,
Computer Graphics (SIGGRAPH 92),
Chicago, Illinois, July, 1992, p 379-387.

Fowler et al. Figure 7

Example: Seashells

- Sweep generating curve around helico-spiral axis



Example: Seashells

- Sweep generating curve around helico-spiral axis

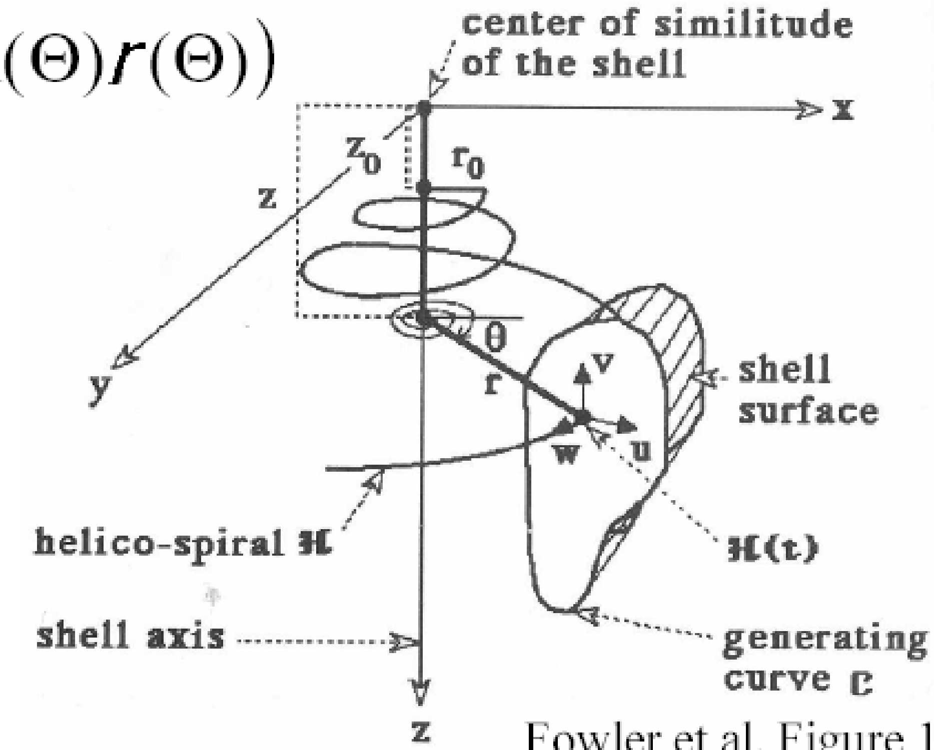
Helico-Spiral definition:

$$H(\Theta) = \left(\cos(\Theta)r(\Theta), z(\Theta), \sin(\Theta)r(\Theta)\right)$$

Θ (angle)

$r(\Theta) = e^{\lambda\Theta}$ (radius)

$z(\Theta) = e^{\mu\Theta}$ (height)



Fowler et al. Figure 1

Example: Seashells

- Sweep generating curve around helico-spiral axis

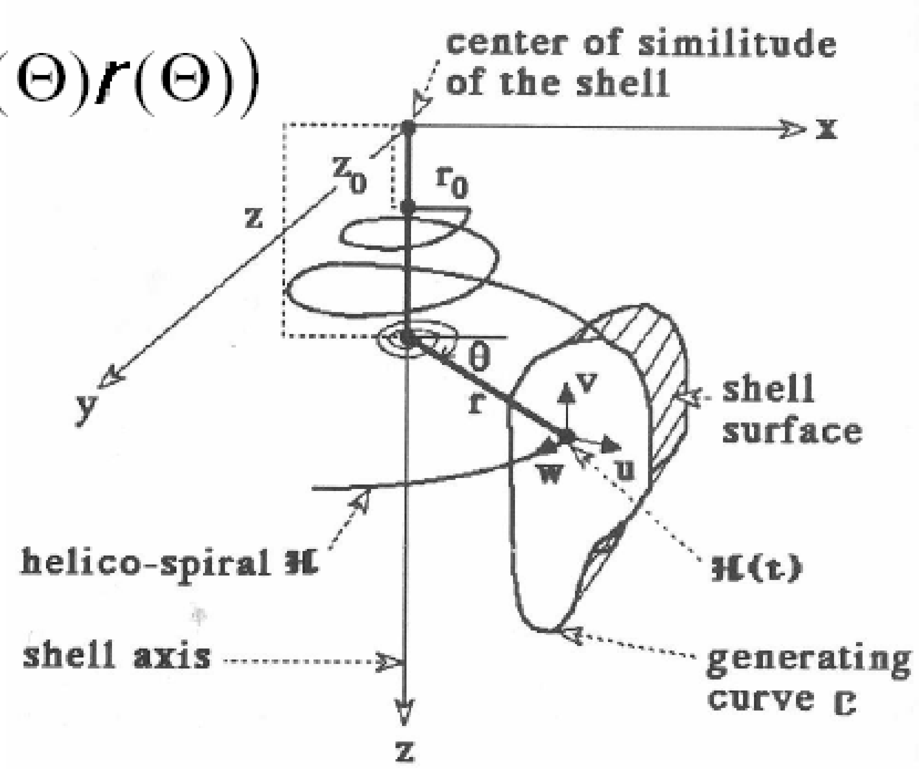
Helico-Spiral definition:

$$H(\Theta) = \left(\cos(\Theta) r(\Theta), z(\Theta), \sin(\Theta) r(\Theta)\right)$$

Θ (angle)

$r(\Theta) = e^{\lambda\Theta}$ (radius)

$z(\Theta) = e^{u\Theta}$ (height)



Shell-Surface Definition:

$$S(\Theta, \Psi) = H(\Theta) + \left(u(\Theta) C_x(\Psi) + v(\Theta) C_y(\Psi)\right) r(\Theta)$$

Example: Seashells

- Sweep generating curve around helico-spiral axis

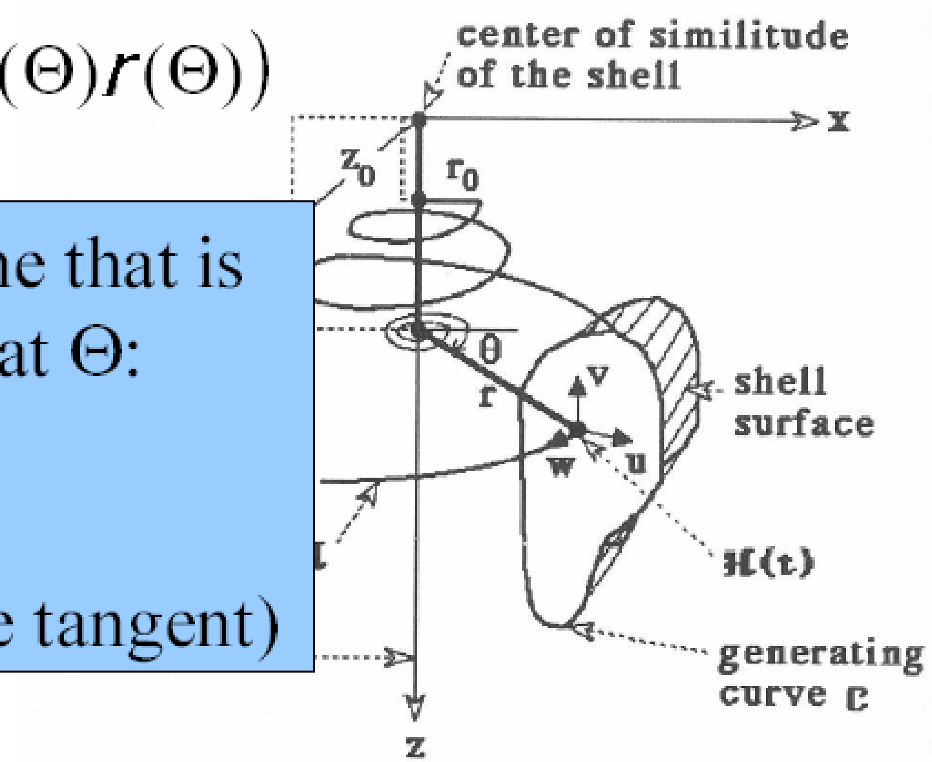
Helico-Spiral definition:

$$H(\Theta) = \big(\cos(\Theta)r(\Theta), z(\Theta), \sin(\Theta)r(\Theta)\big)$$

Θ (angle)

$u(\Theta)$ and $v(\Theta)$ define the plane that is perpendicular to the curve H at Θ:

- $u(\Theta)$ is the curve <u>normal</u>
- $v(\Theta)$ is the curve <u>bi-tangent</u> (perp. to $u(\Theta)$ and the curve tangent)



Shell-Surface Definition:

$$S(\Theta,\Psi) = H(\Theta) + \big(u(\Theta)C_x(\Psi) + v(\Theta)C_y(\Psi)\big)r(\Theta)$$

Example: Seashells

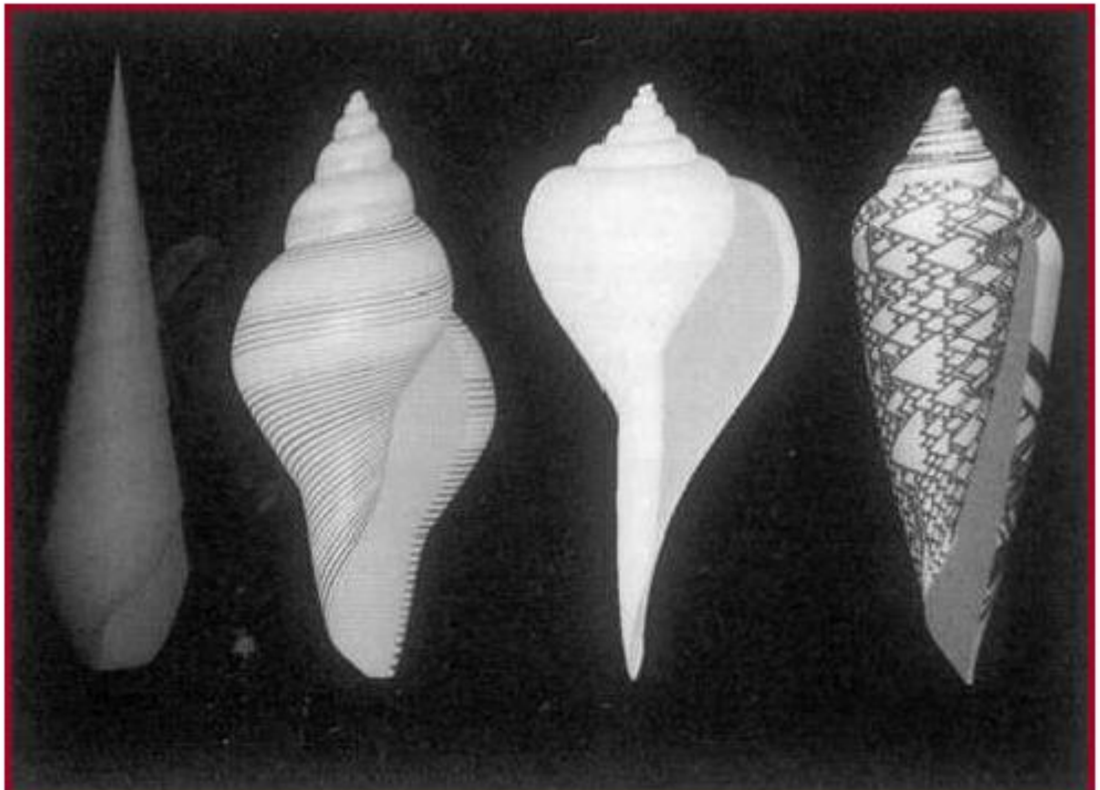
- Generate different shells by varying parameters

Different helico-spirals

Fowler et al. Figure 2

Example: Seashells

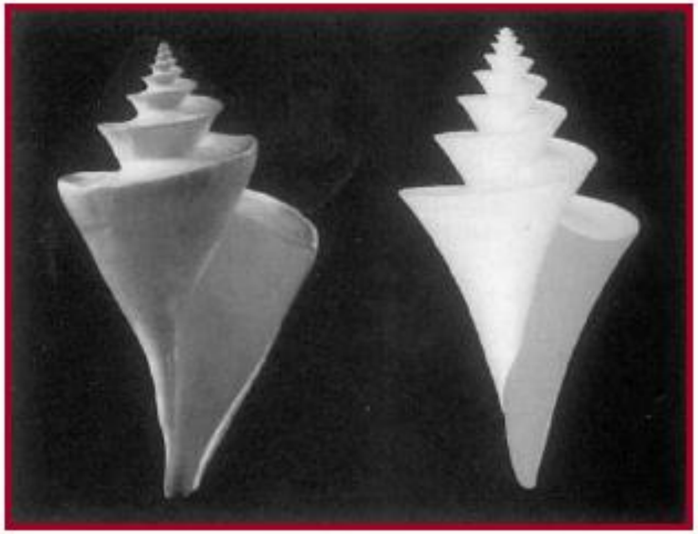
- Generate different shells by varying parameters



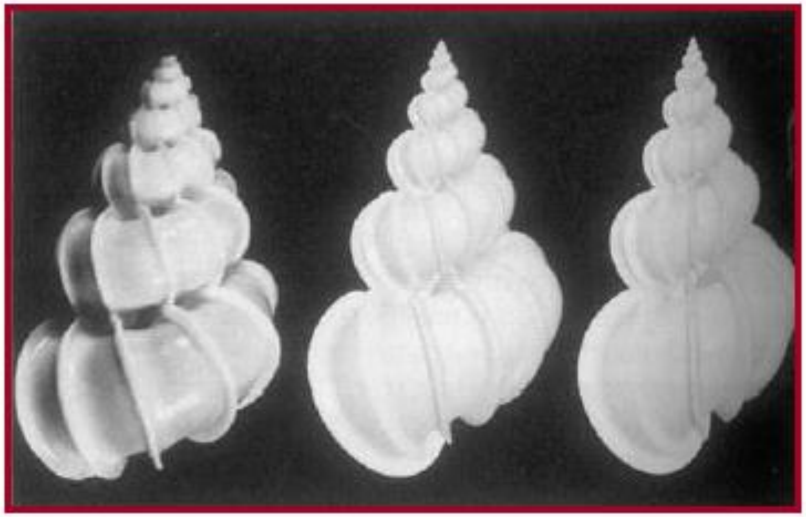
Different generating curves

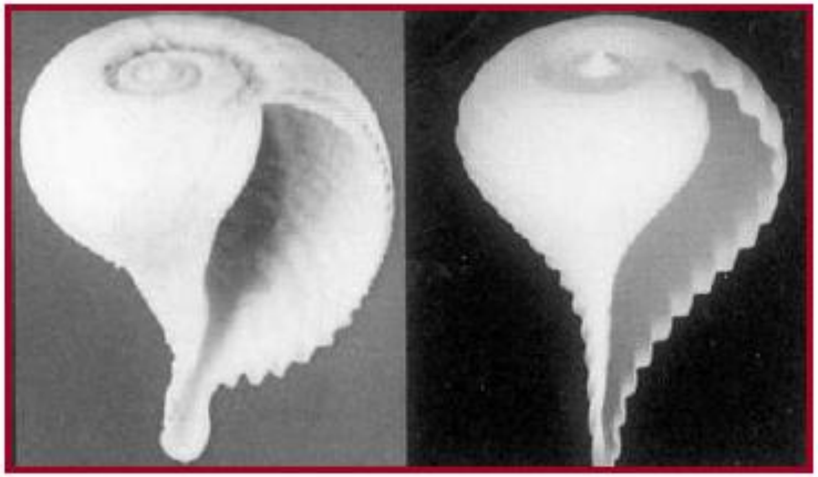
Fowler et al. Figure 3

Example: Seashells



Generate many interesting shells with a simple procedural model!

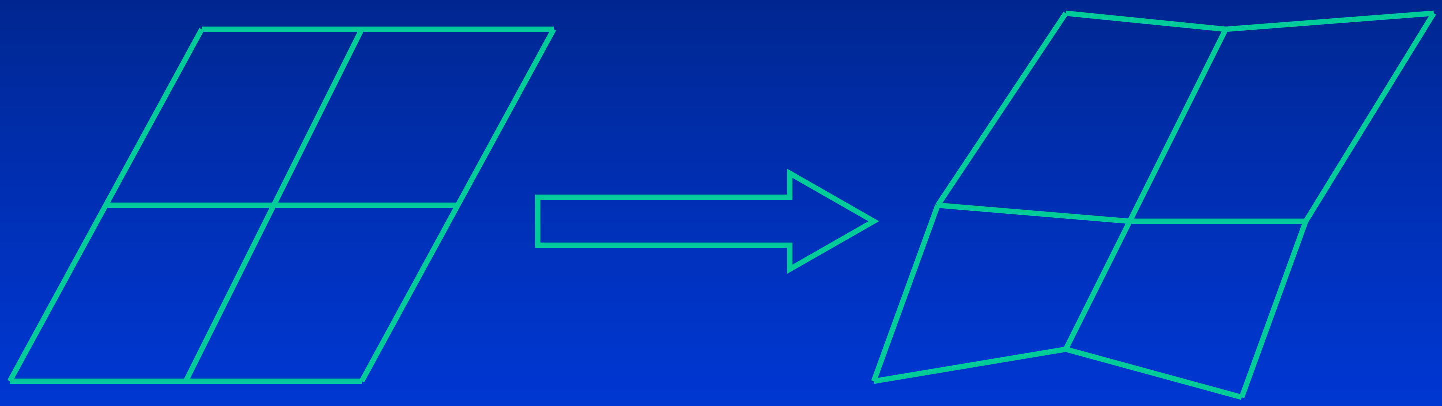




Fowler et al. Figures 4,5,7

Procedural Terrain

- **"Subdivide and displace"**

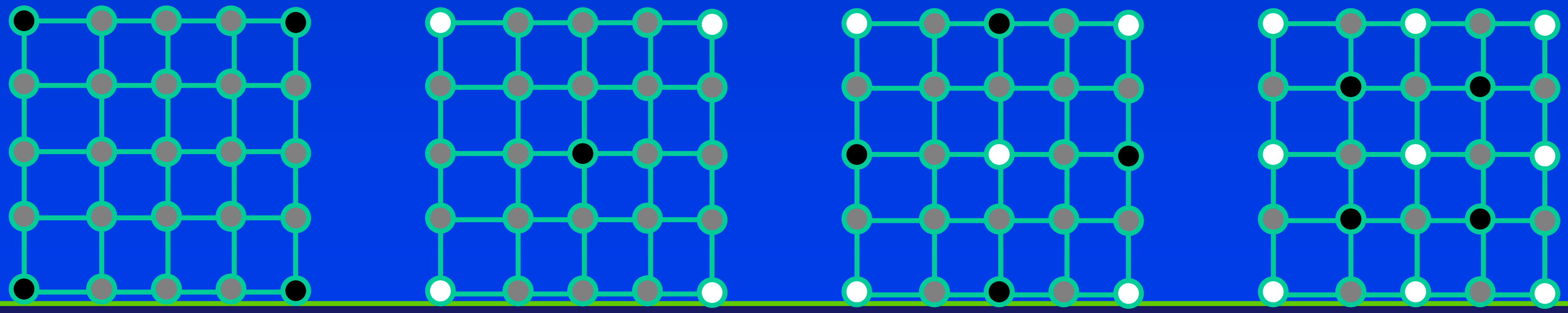


Simple Explicit Procedural Model

- **Begin with a regular mesh**
- **Perturb vertex geometry procedurally (typically pseudo-randomly)**
- **Iterate this process until desired shape is achieved**
- **Very general technique that can also be used to add irregularity (“noise”) to arbitrary mesh objects**

Midpoint Displacement For Terrain

- **Seed corners with values**
- **Perturb midpoint randomly from mean**
- **Recursion using a smaller window**
- **In 2D, best to use "diamond-square" recursion (to prevent axis-aligned artifacts)**

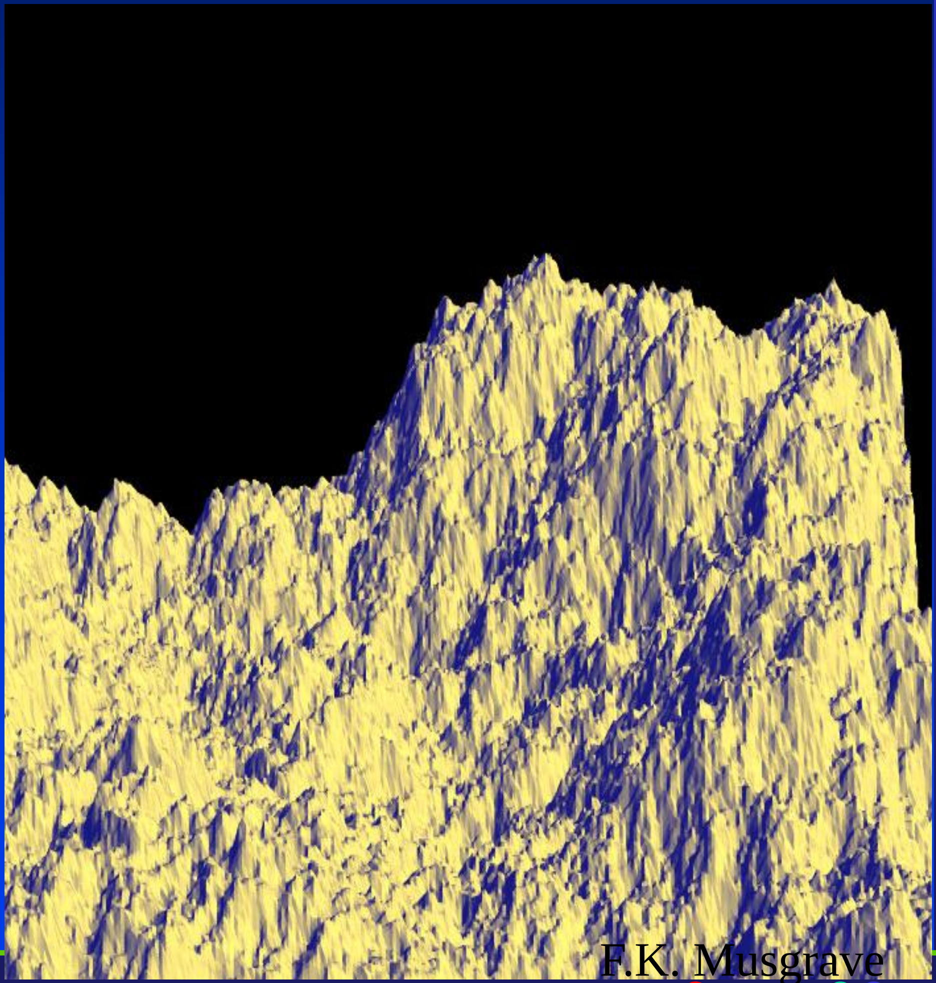


One Example: Natural Terrain Modeling

Fractal Noise Terrain

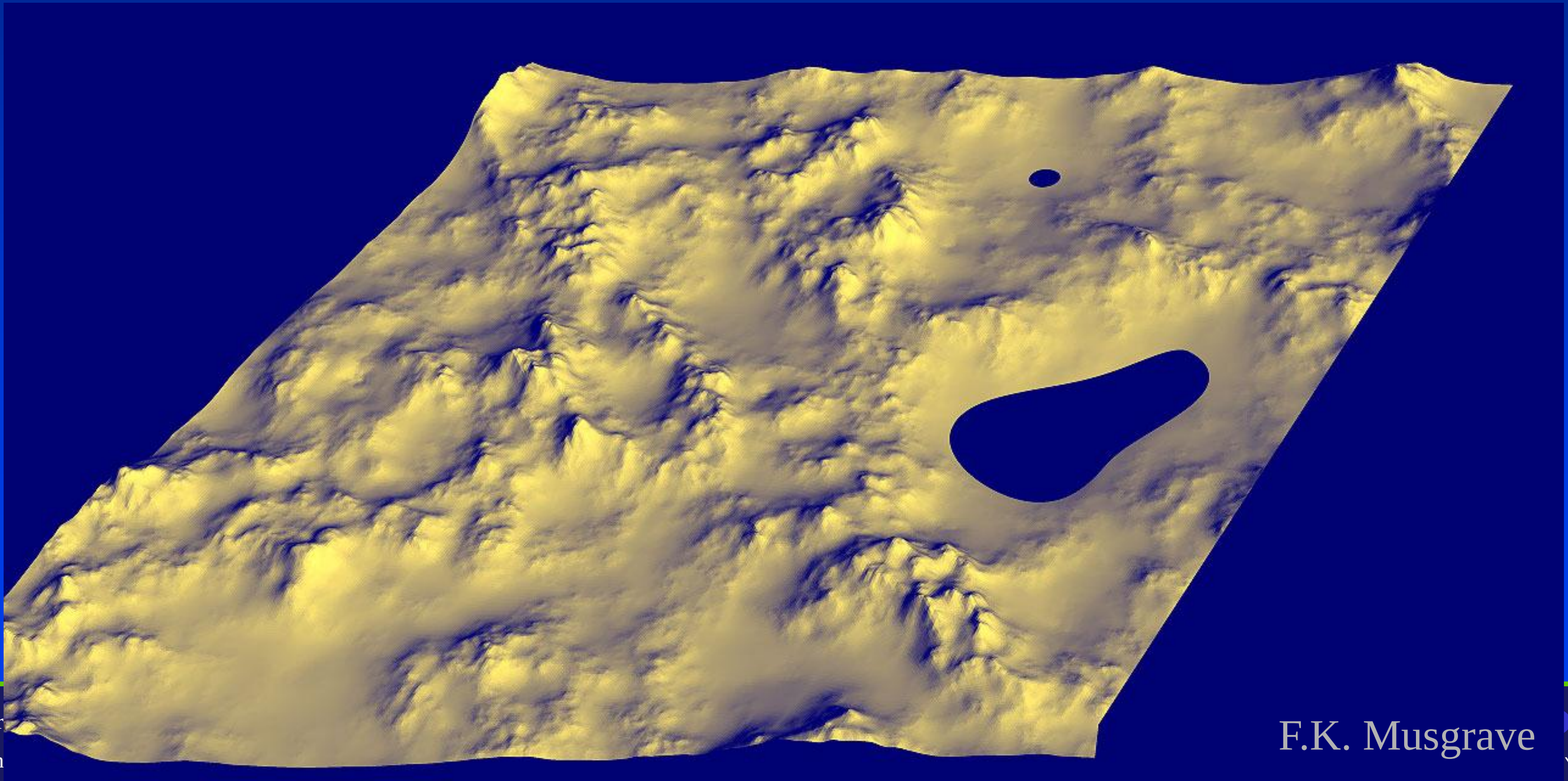
- Use fractal noise to generate terrain
- Can be made tile-able over unit square:

$$F_{tileable}(x,y) = [$$
$$F(x,y) * (1-x) * (1-y) +$$
$$F(x-1,y) * x * (1-y) +$$
$$F(x-1,y-1) * x * y +$$
$$F(x,y-1) * (1-x) * y]$$



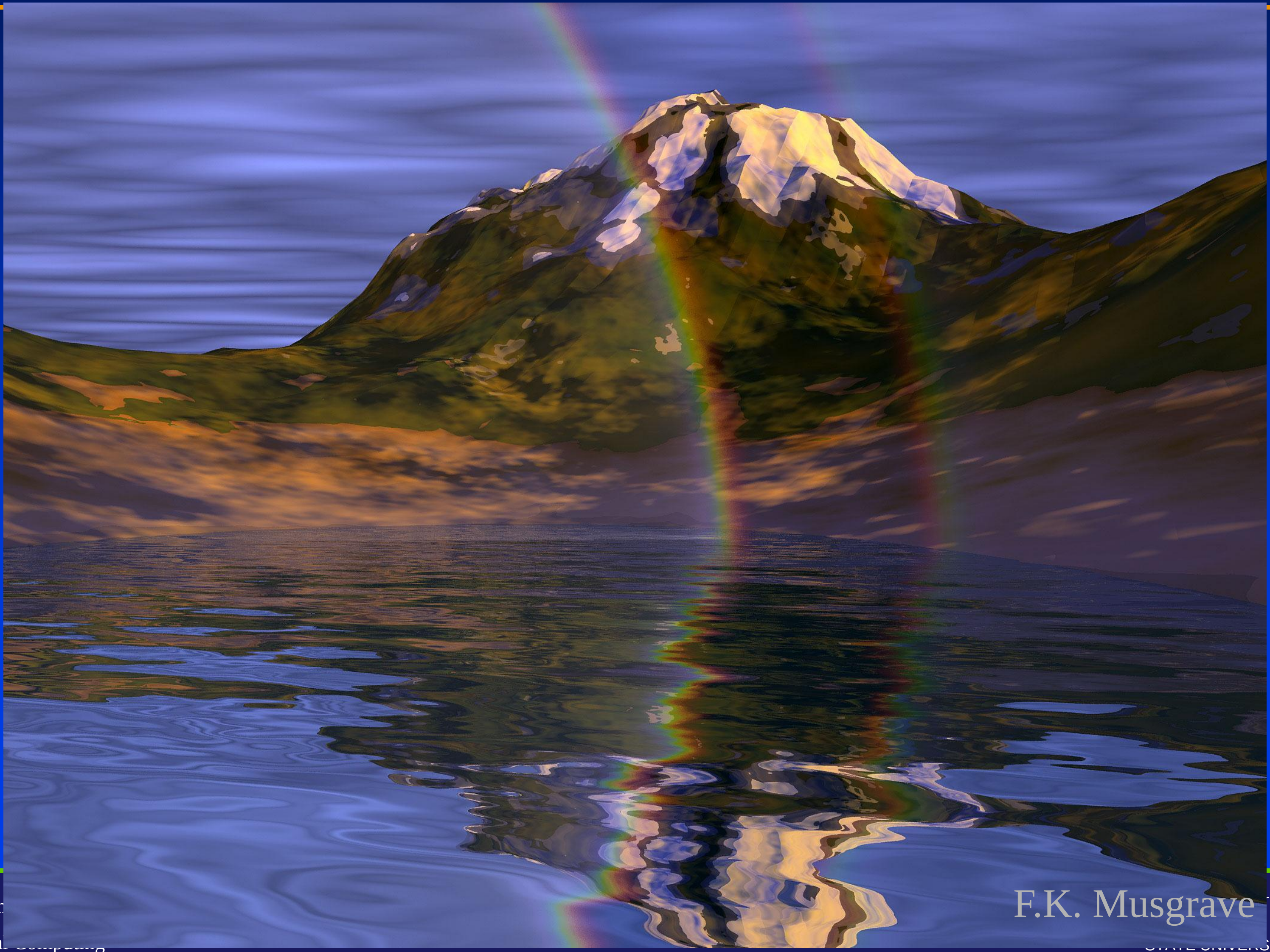
Adding Water

- **Use an elevation threshold ($z < z_{water}$)**



Terrain Example

Terrain Example



Terrain Example

Terrain Example

Terragen

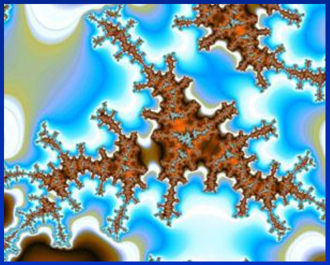
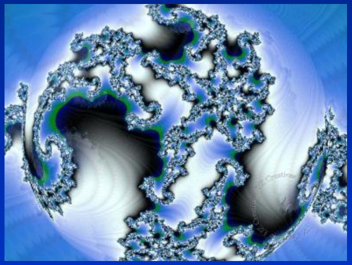
- **Commercial product (free for personal use)**
- **Website: http://www.planetside.co.uk/terragen/**
- **This image took ~3 minutes to set up**

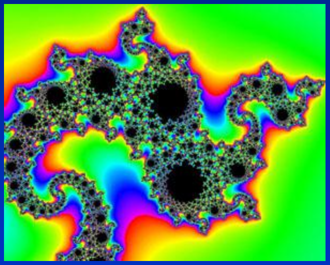
Procedural Modeling

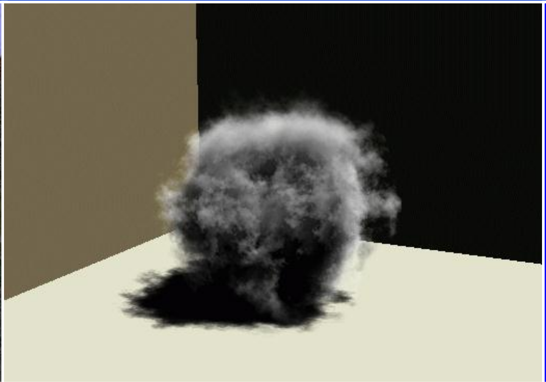
- Sweeps
- Fractals
- Grammars

Fractal Geometry

- All of the modeling techniques covered so far use Euclidean geometry methods
  - Objects were described using equations
- This is fine for manufactured objects
- But what about natural objects that have irregular or fragmented features?
  - Mountains, clouds, coral…



"Clouds are not spheres, mountains are not cones, coastlines are not circles and bark is not smooth, nor does lightning travel in a straight line."

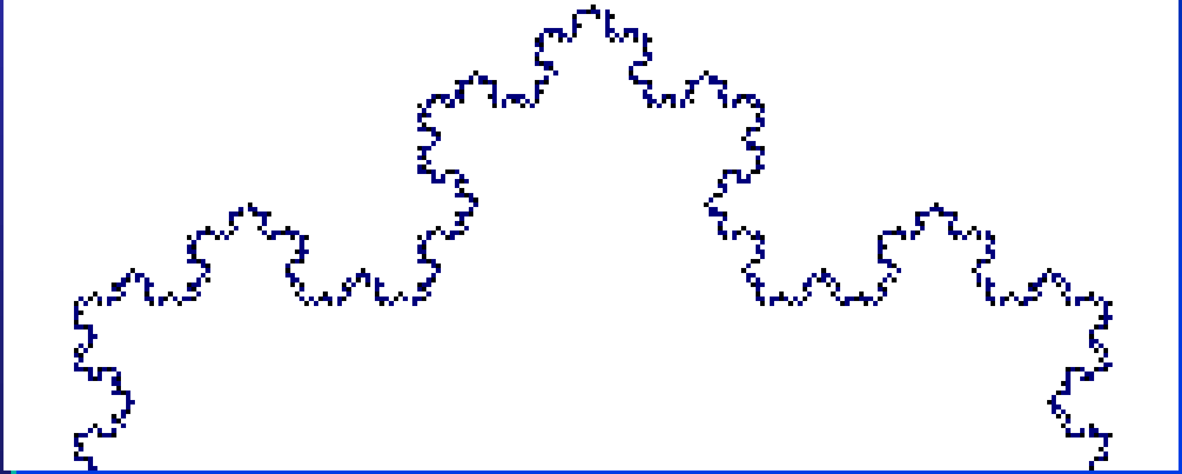
Benoit Mandelbrot

Fractal Geometry and Procedural Modeling

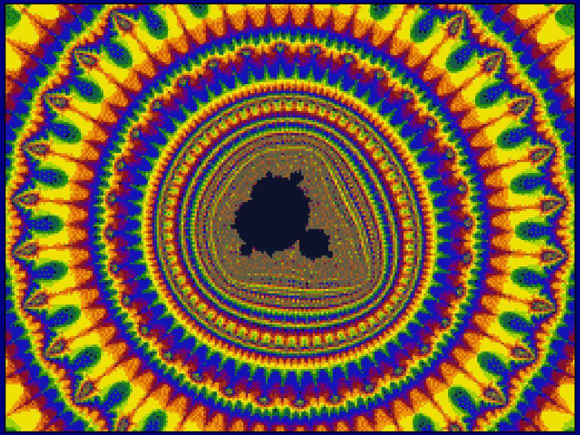
- Natural objects can be realistically described using **fractal geometry methods**
- Fractal methods use procedures rather than equations to model objects - **procedural modeling**
- The major characteristic of any procedural model is that the model is not based on data, but rather on the implementation of a procedure following a particular set of rules

Modeling on the Fly!!!

- A fractal object has two basic characteristics:
  - Infinite detail at every point
  - A certain self similarity between object parts and the overall features of the object



The Koch Curve



Mandelbrot Set Video From:

http://www.fractal-animation.net/ufvp.htm

Fractal Generator

- A fractal object is generated by repeatedly applying a specified transform function to points in a region of space
- If $P_0 = (x_0, y_0, z_0)$ is a selected initial position, each iteration of a transformation function F generates successive levels of detail with the calculations:

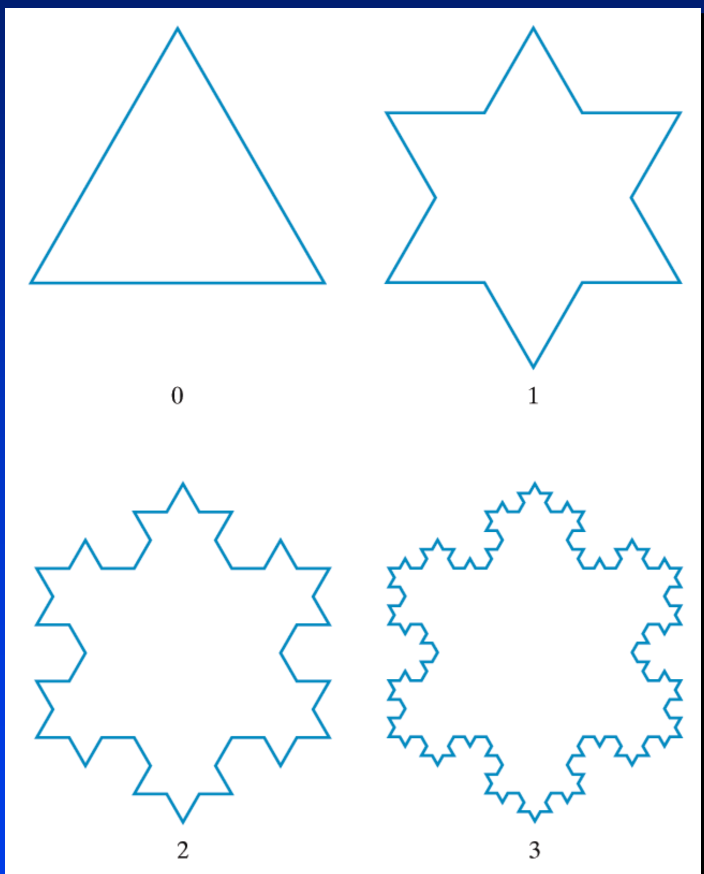
$$P_1 = F(P_0), \quad P_2 = F(P_1), \quad P_3 = F(P_2), \quad \ldots$$

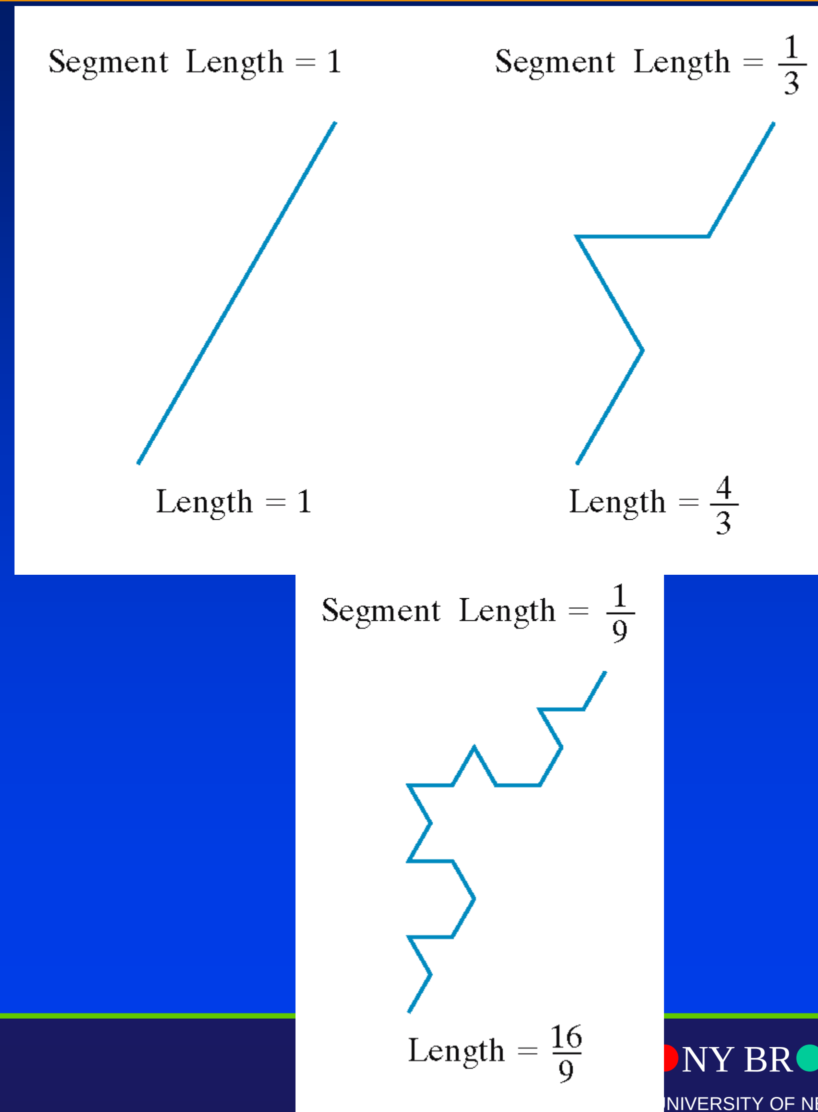
- In general the transformation is applied to a specified point set, or to a set of primitives (e.g. lines, curves, surfaces)

Fractal Generator

- Although fractal objects, by definition have infinite detail, we only apply the transformation a finite number of times
- Obviously objects we display have finite dimension – they fit on a page or a screen
- A procedural representation approaches a *true* representation as we increase the number of iterations
- The amount of detail is limited by the resolution of the display device, but we can always *zoom in* for further detail

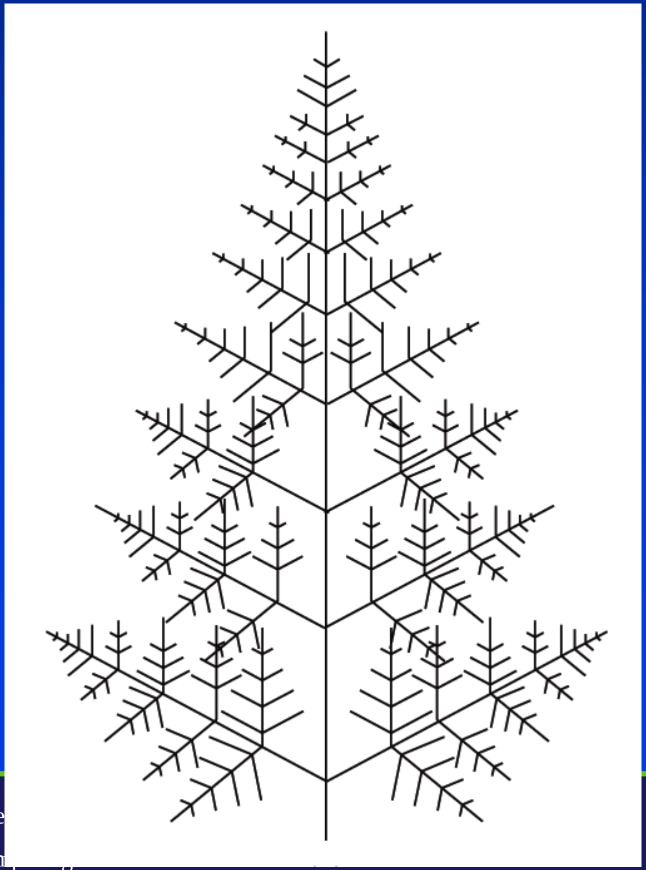
The Koch Snowflake

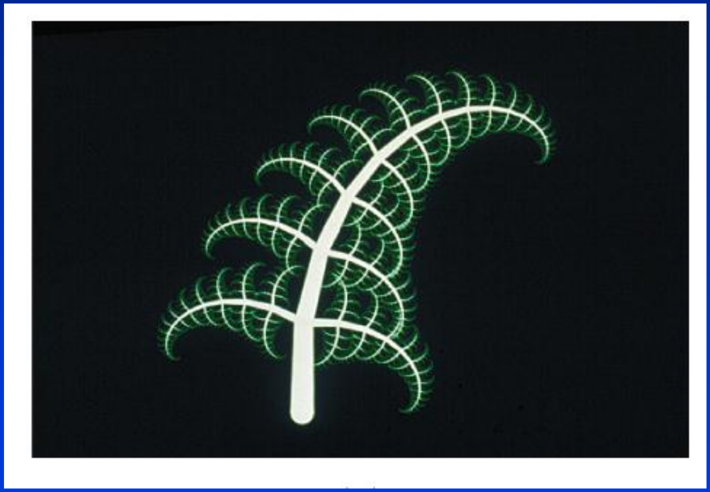




Example: Ferns

- Very similar techniques can be used to generate vegetation





Fractal Dimension

- The amount of variation in the structure of a fractal object is described as the **fractal dimension**, D
  - More jagged looking objects have larger fractal dimensions
- Calculating the fractal dimension can be difficult, especially for particularly complex fractals

•We won't look at the details of these calculations

Types of Fractals

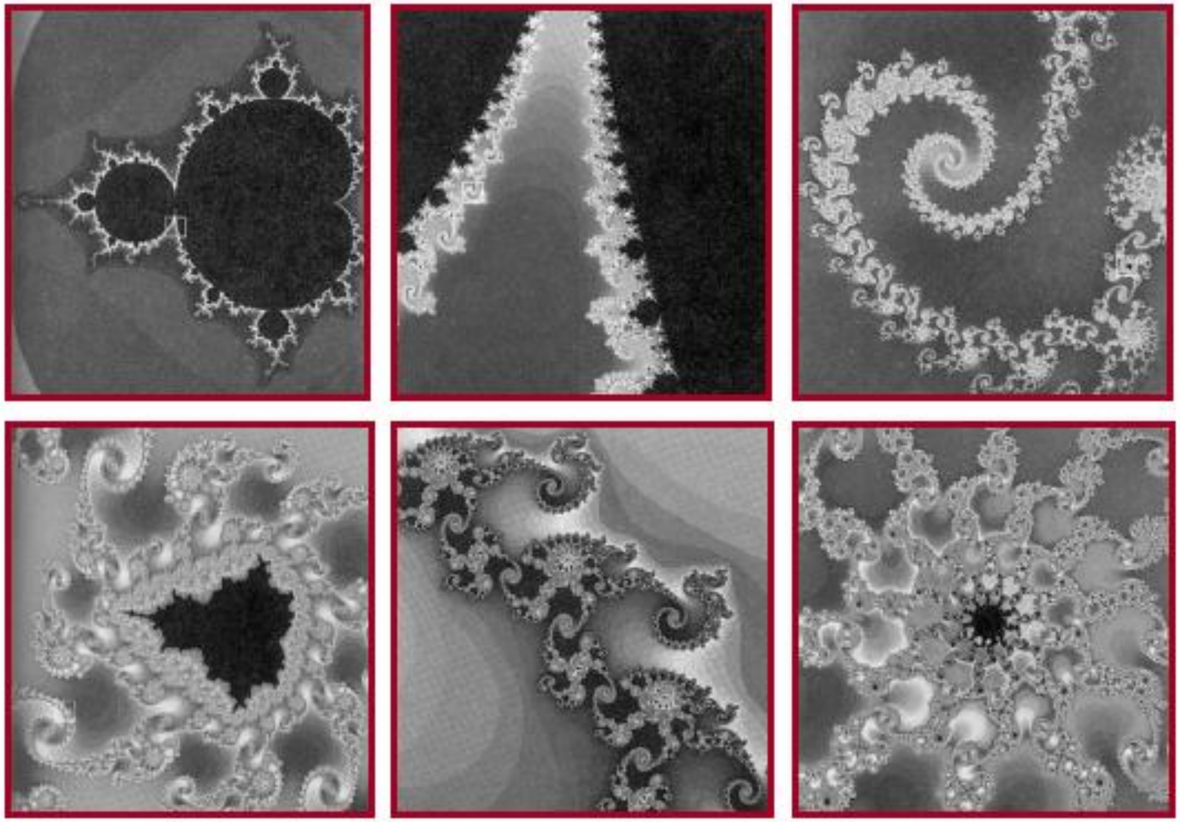
- Fractals can be classified into three groups
  - Self similar fractals
    - These have parts that are scaled down versions of the entire object
    - Commonly used to model trees, shrubs, etc.
  - Self affine fractals
    - Have parts that are formed with different scaling parameters in each dimension
    - Typically used for terrain, water and clouds
  - Invariant fractal sets
    - Fractals formed with non-linear transformations
    - Mandelbrot set, Julia set – generally not so useful

Fractals

- Mandelbrot set
  - $$z_0 = z$$ $$z_k = z_{k-1}^2 + z_0 \qquad k = 1,2,3\ldots$$
  - The boundary of the convergence region in the complex plane is fractal
  - To speed up, we use different color values according to the number of iterations executed by the loop.
  - Could zoom in/out of any particular regions

Fractals

- Defining property:
  - Self-similar with infinite resolution



Mandelbrot Set

H&B Figure 10.100

Fractals

- Useful for describing natural 3D phenomenon
  - Terrain
  - Plants
  - Clouds
  - Water
  - Feathers
  - Fur
  - etc.

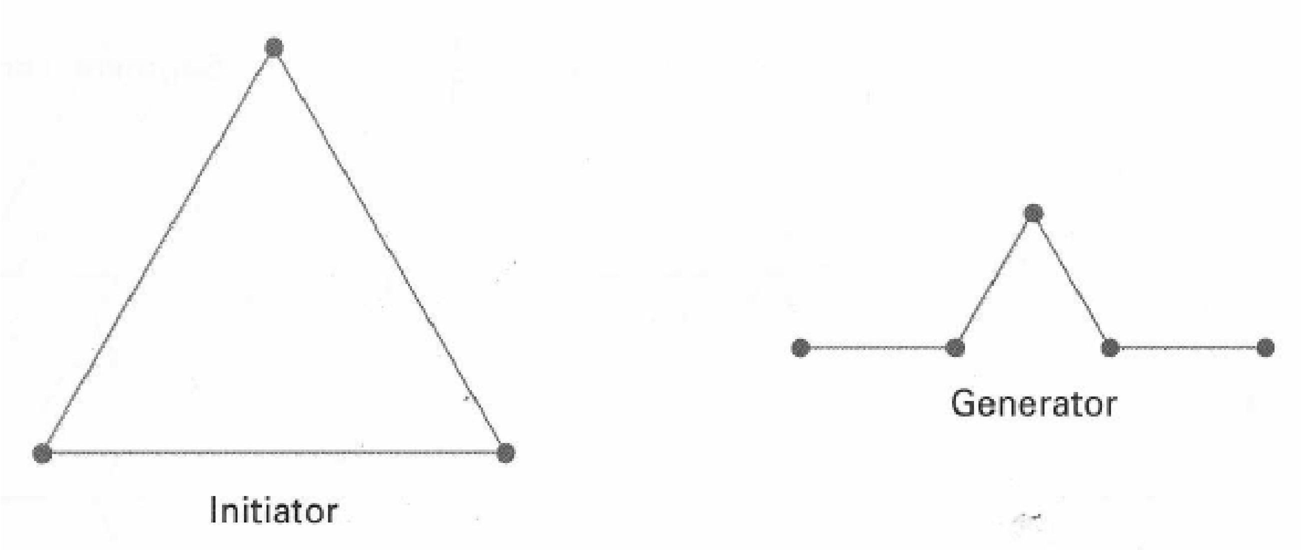
H&B Figure 10.80

Fractal Generation

- Deterministically self-similar fractals
  - Parts are scaled copies of original
- Statistically self-similar fractals
  - Parts have same statistical properties as original

Deterministic Fractal Generation

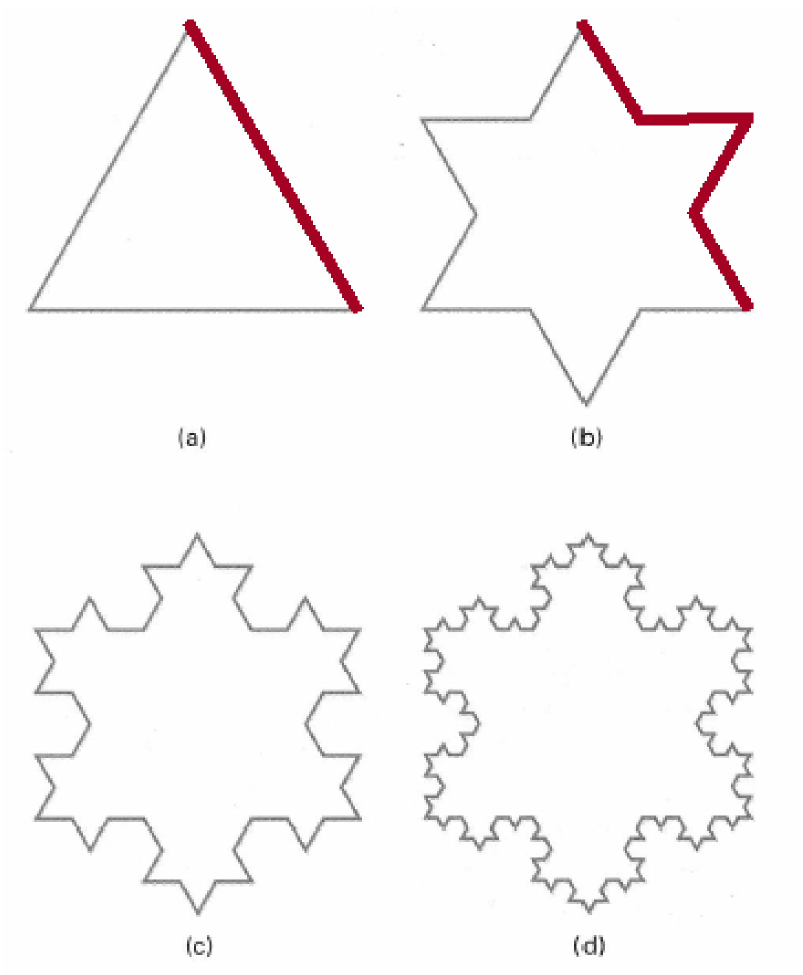
- General procedure:
  - Initiator: start with a shape
  - Generator: replace subparts with scaled copy of original



H&B Figure 10.68

Deterministic Fractal Generation

- Apply generator repeatedly

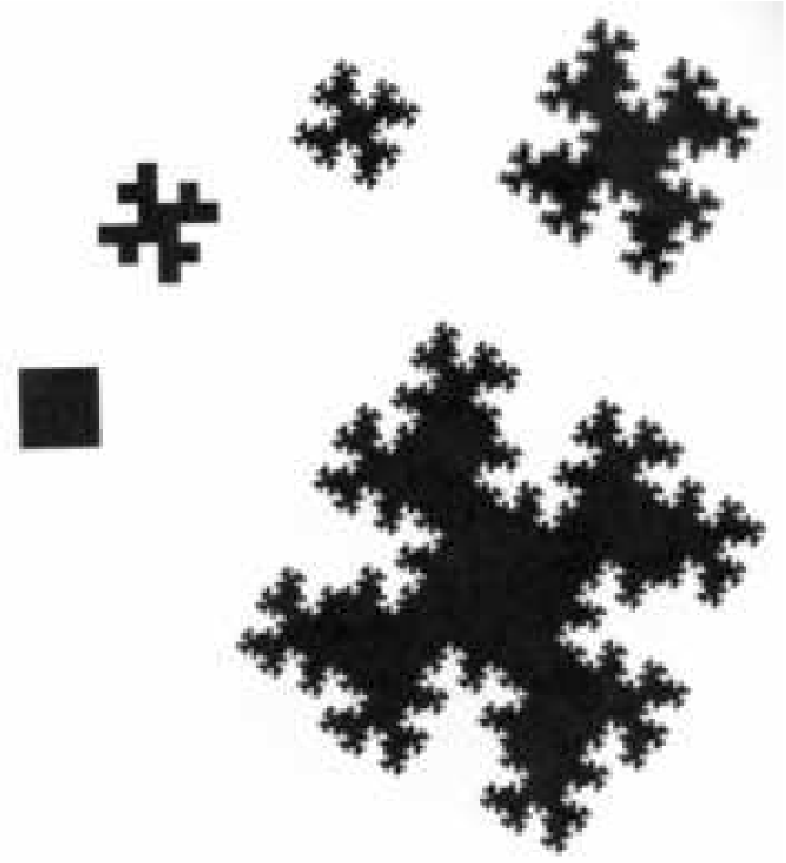


Koch Curve

H&B Figure 10.69

Deterministic Fractal Generation

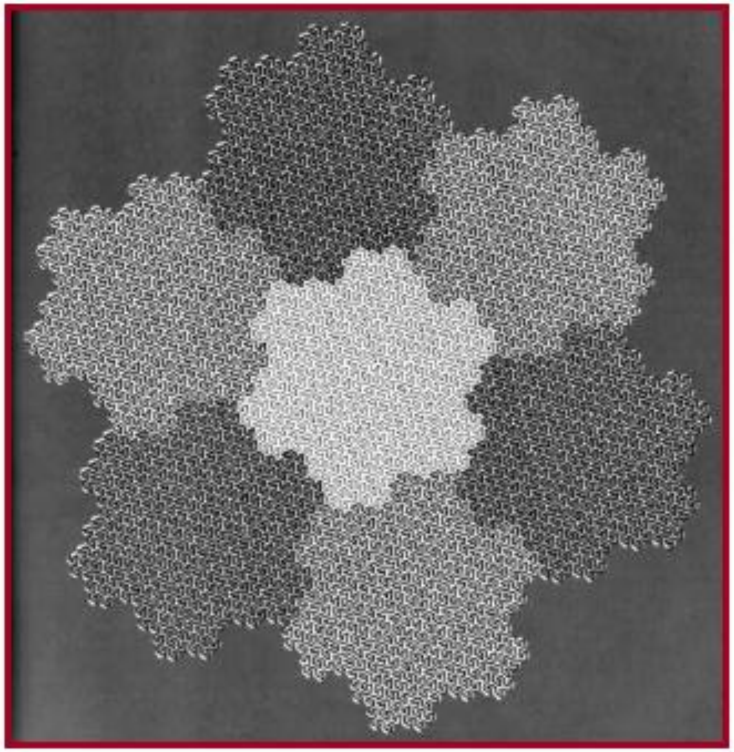
- Useful for creating interesting shapes!



Mandelbrot Figure X

Deterministic Fractal Generation

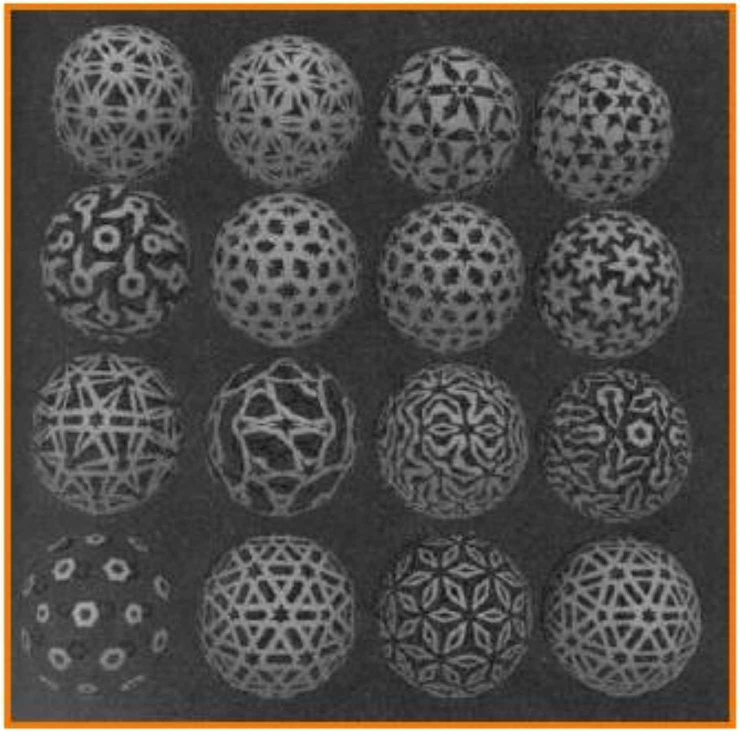
- Useful for creating interesting shapes!

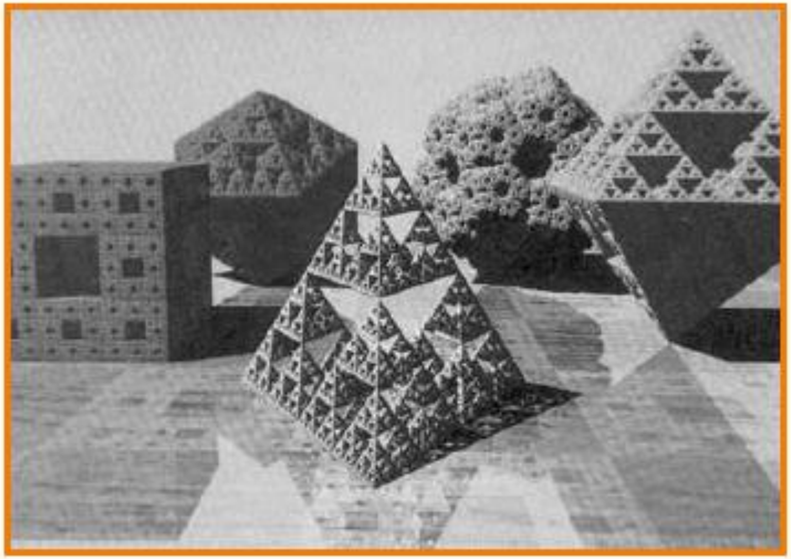


Mandelbrot Figure 46

Deterministic Fractal Generation

- Useful for creating interesting shapes!





H&B Figures 75 & 109

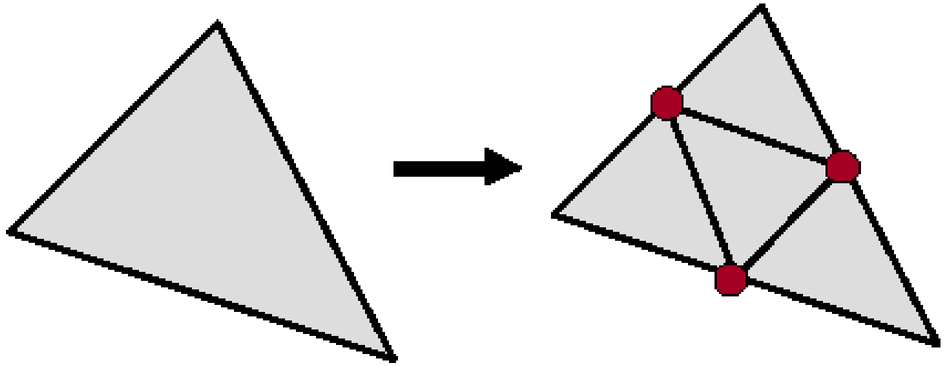
Fractal Generation

- Deterministically self-similar fractals
  - Parts are scaled copies of original

- Statistically self-similar fractals
  - Parts have same statistical properties as original

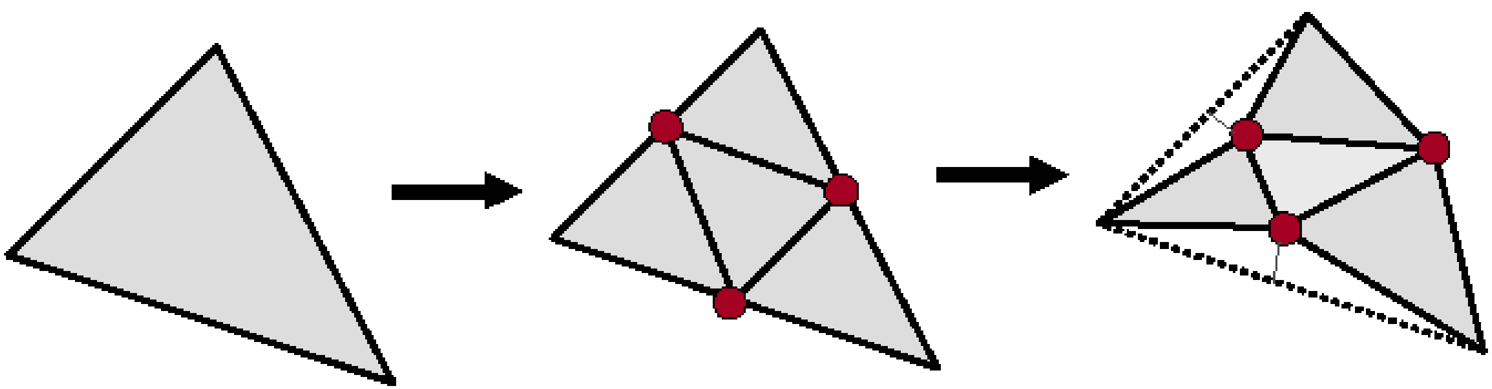
Statistical Fractal Generation

- General procedure:
  - Initiator: start with a shape
  - Generator: replace subparts with a self-similar



Statistical Fractal Generation

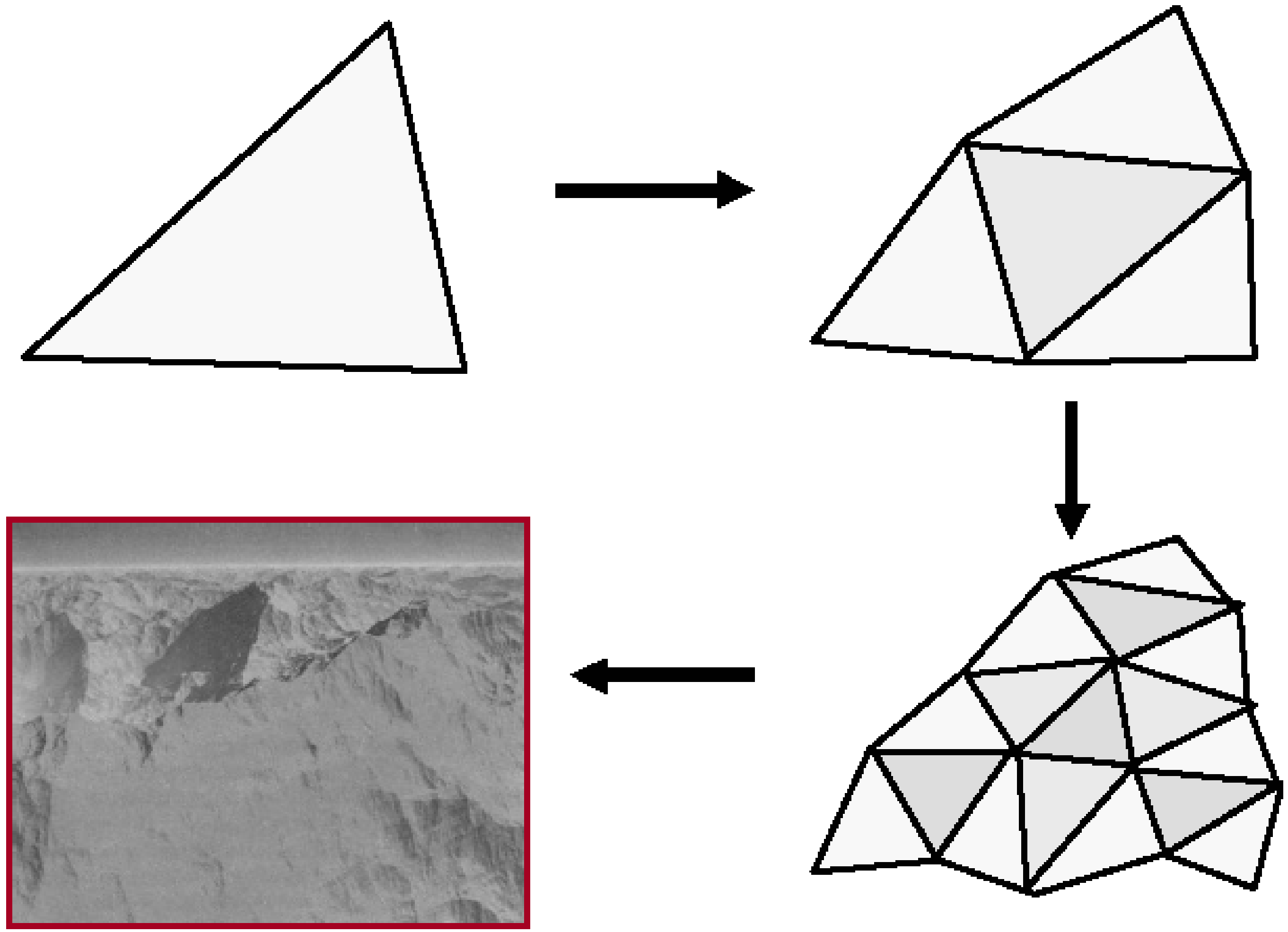
- General procedure:
  - Initiator: start with a shape
  - Generator: replace subparts with a self-similar random pattern



Random Midpoint Displacement

Statistical Fractal Generation

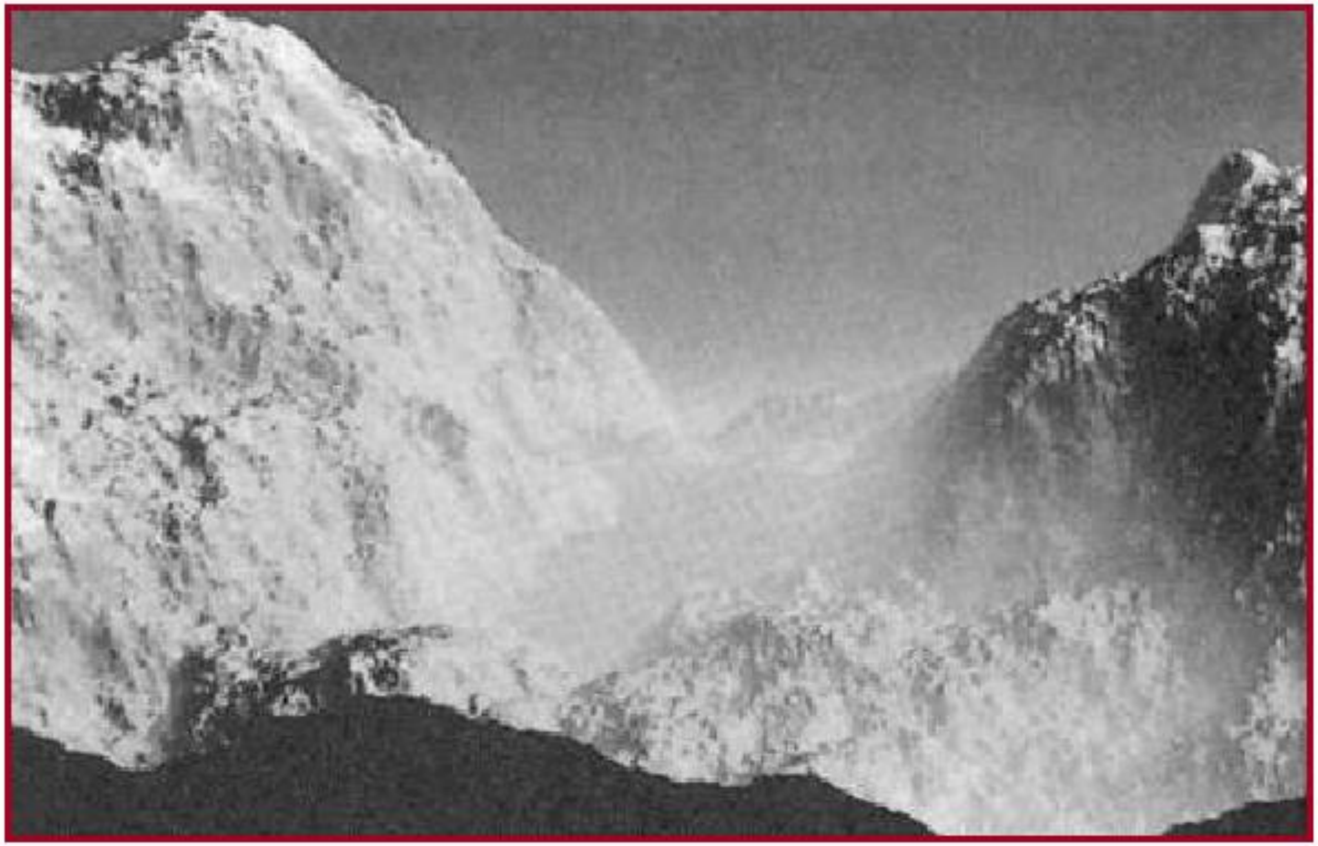
- Example: terrain



H&B Figure 10.83b

Statistical Fractal Generation

- Useful for creating mountains



H&B Figure 10.83a

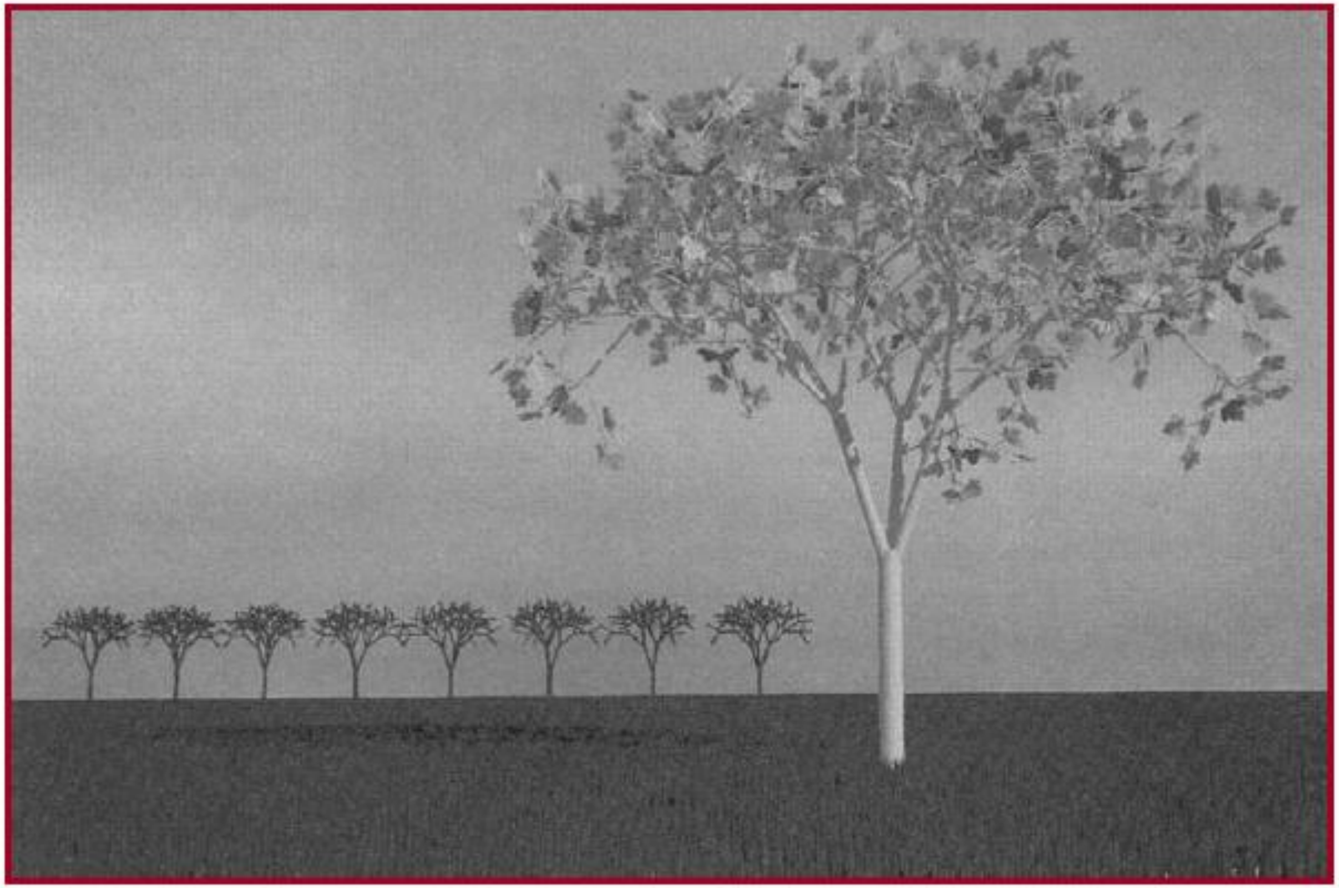
Statistical Fractal Generation

- Useful for creating 3D plants

H&B Figure 10.82

Statistical Fractal Generation

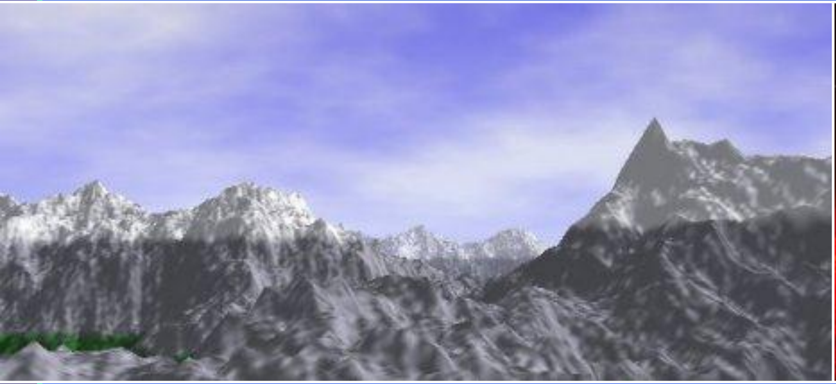
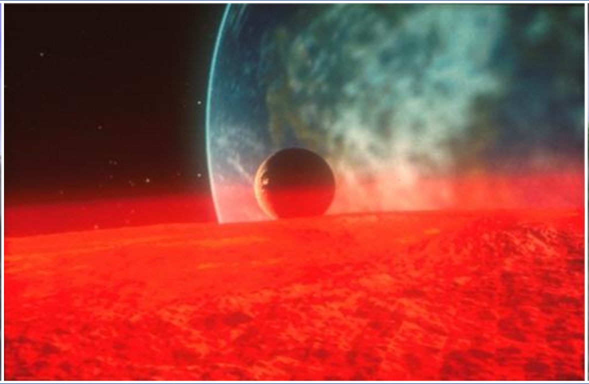
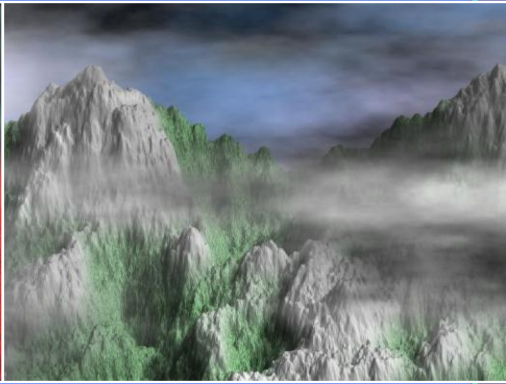
- Useful for creating 3D plants



H&B Figure 10.79

Random Midpoint Displacement Methods for Topography

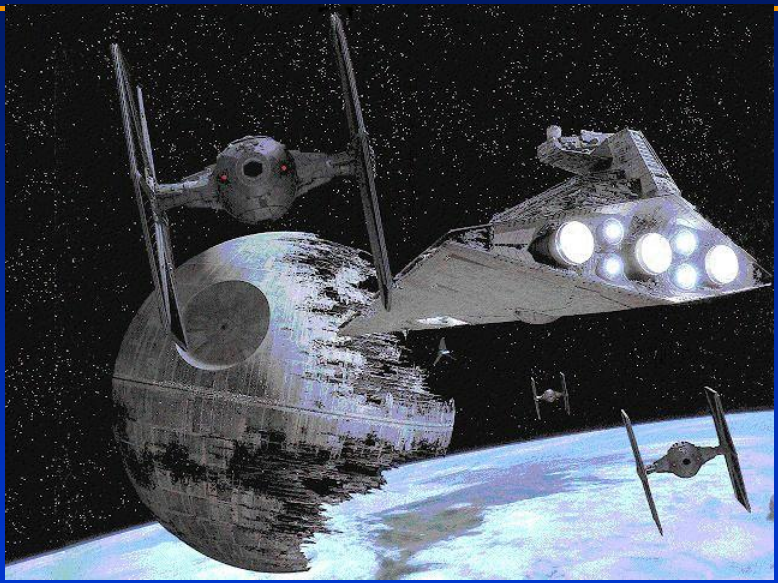
- One of the most successful uses of fractal techniques in graphics is the generation of landscapes
- One efficient method for doing this is **random midpoint displacement**

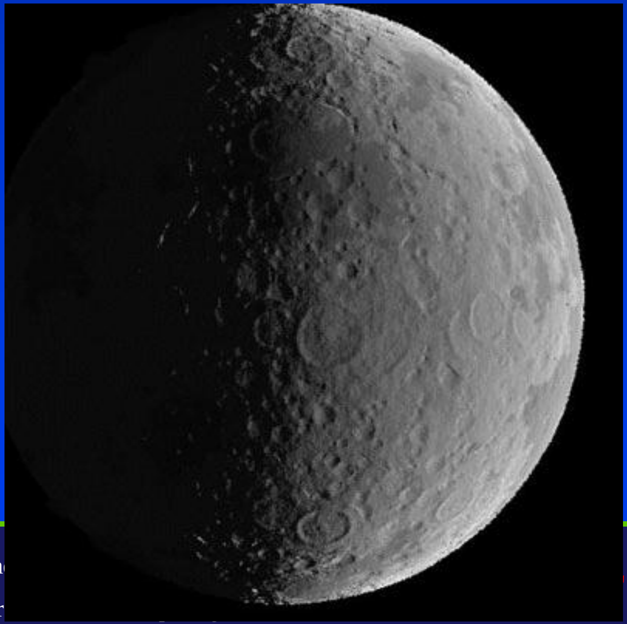




Random Midpoint Displacement Methods for Topography

- Easy to do in two dimensions
- Easily expanded to three dimensions to generate terrain
- Can introduce a roughness factor H to control terrain appearance
- Control surfaces can be used to start with a general terrain shape

Fractals in Film Special Effects





Fractal Summary

- Fractals in particular are a fairly exotic modelling technique, but can be extremely effective

Database Amplification

- Procedure-based digital content generation is very attractive because it allows for significant database amplification
- Limited input data produces rich & a large variety of output forms
  - E.g., Perlin noise function + basic math gives fire, clouds, wood, etc.
- If it can be generated on the fly…
  - Artist doesn’t have to design it
  - Don’t need to store/transmit it

Procedural Modeling

L-Systems

Procedural Terrain

Procedural Behavior

L-Systems (Background)

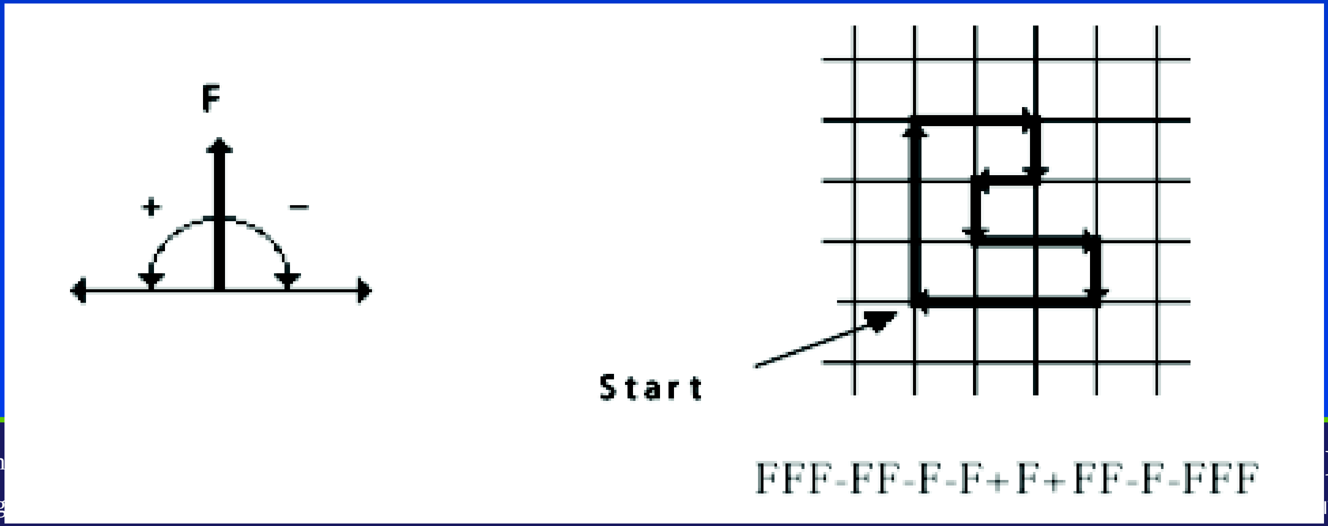
- Developed by Aristid Lindenmayer to model the development of plants
- Based on parallel string-rewriting rules
- Excellent for modeling organic objects and fractals

L-Systems Grammar (Concepts)

- Begin with a set of “productions” (replacement rules) and a “seed” axiom
- In parallel, all matching productions are replaced with their right-hand sides
- Example:
  - Rules:
    - B -> ACA
    - A -> B
  - Axiom: AA
  - Sequence: AA, BB, ACAACA, BCBBCB, etc.
- Strings are converted to graphics representations via interpretation as turtle graphics commands

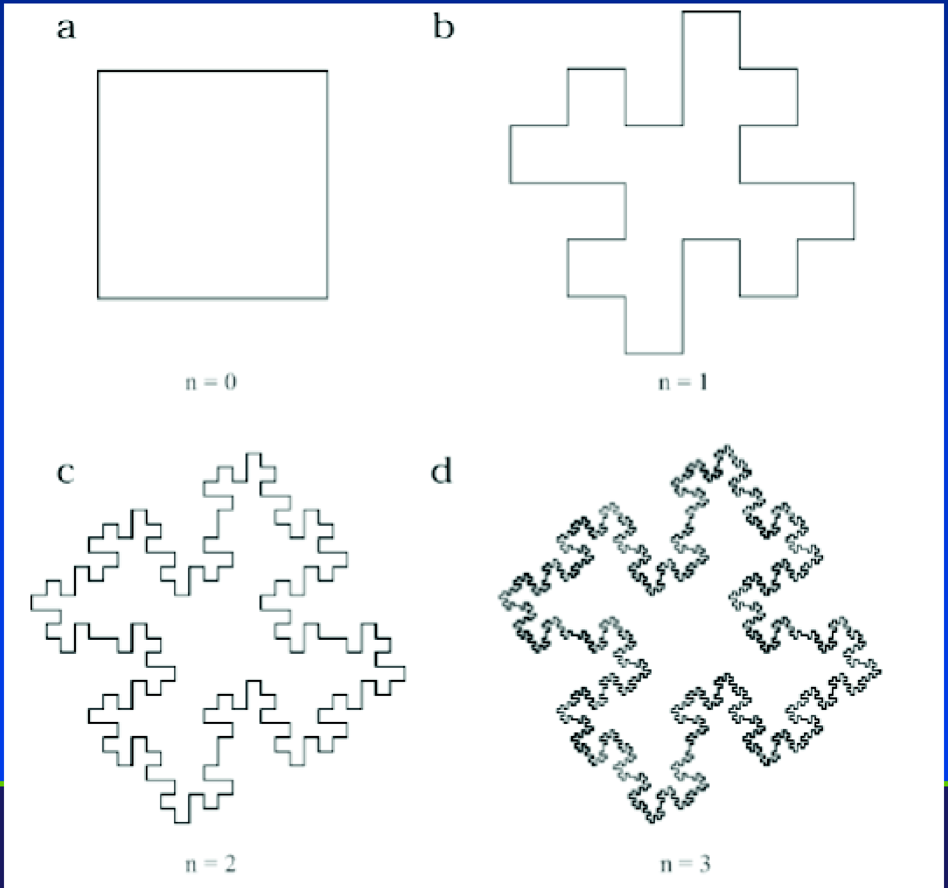
Turtle Commands

- F_x: move forward one step, drawing a line
- f_x: move forward one step, without drawing a line
- $+_x$: turn left by angle ∂
- $-_x$: turn right by angle ∂



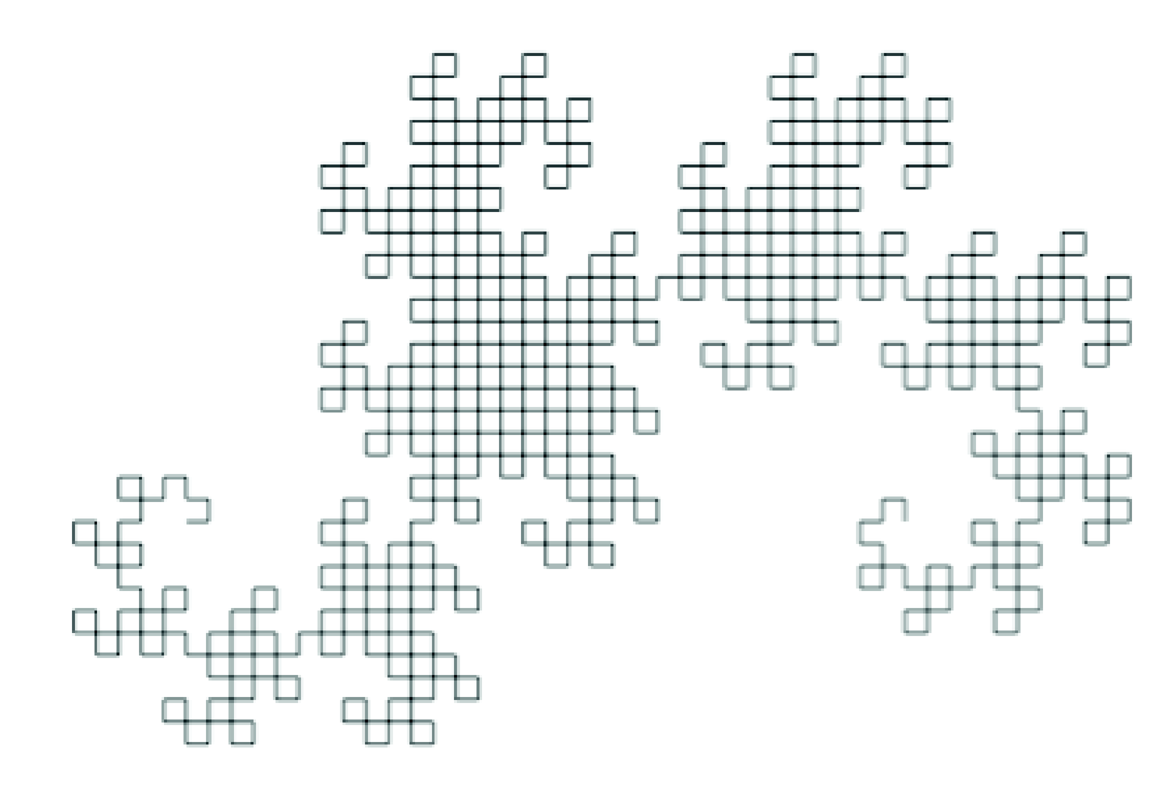
L-Systems Example: Koch Snowflake

- Axiom: F-F-F-F ∂ :90 degrees
- F -> F-F+F+FF-F-F+F



L-Systems Example: Dragon Curve

- Axiom:F_l ∂ :90 degrees n:10 iterations
- F_l -> F_l+F_r+
- F_r -> F_l-F_r-



Department of Computer Science
Center for Visual Computing

L-Systems Grammar: Extensions

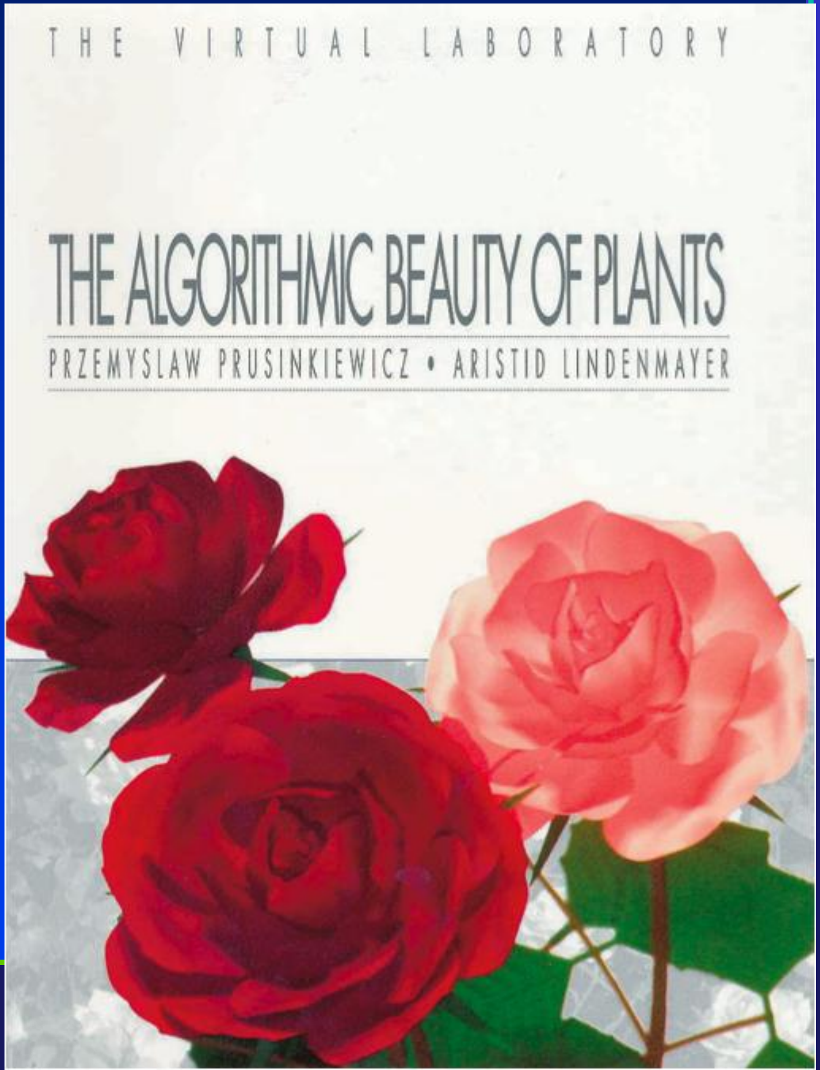
- Basic L-Systems have inspired a large number of variations
- Context sensitive: productions look at neighboring symbols
- Bracketed: save/restore state (for branches)
- Stochastic: choose one of n matching productions randomly
- Parametric: variables can be passed between productions

L-Systems for Plants

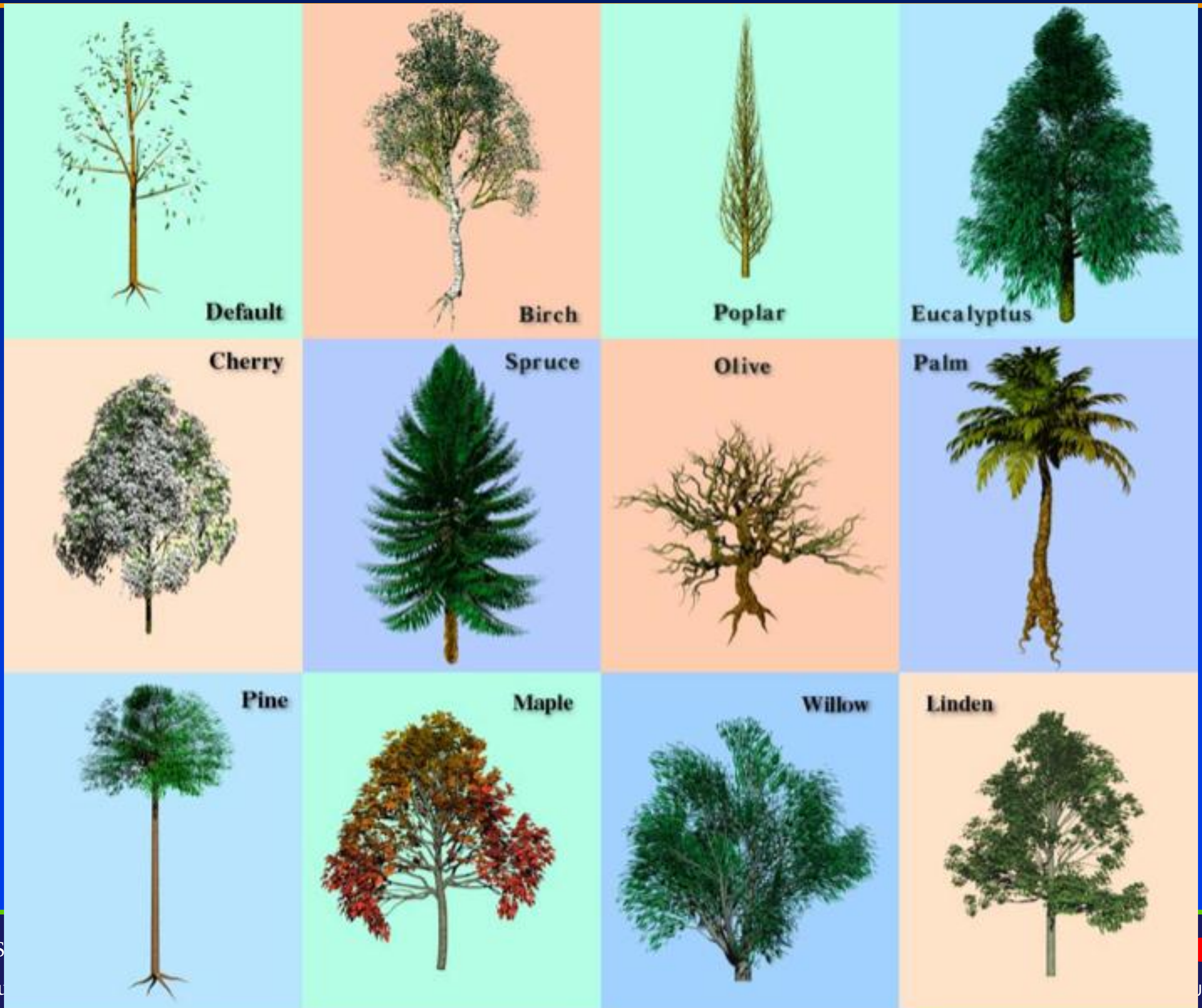
- L-Systems can capture a large array of plant species
- Designing rules for a specific species can be challenging

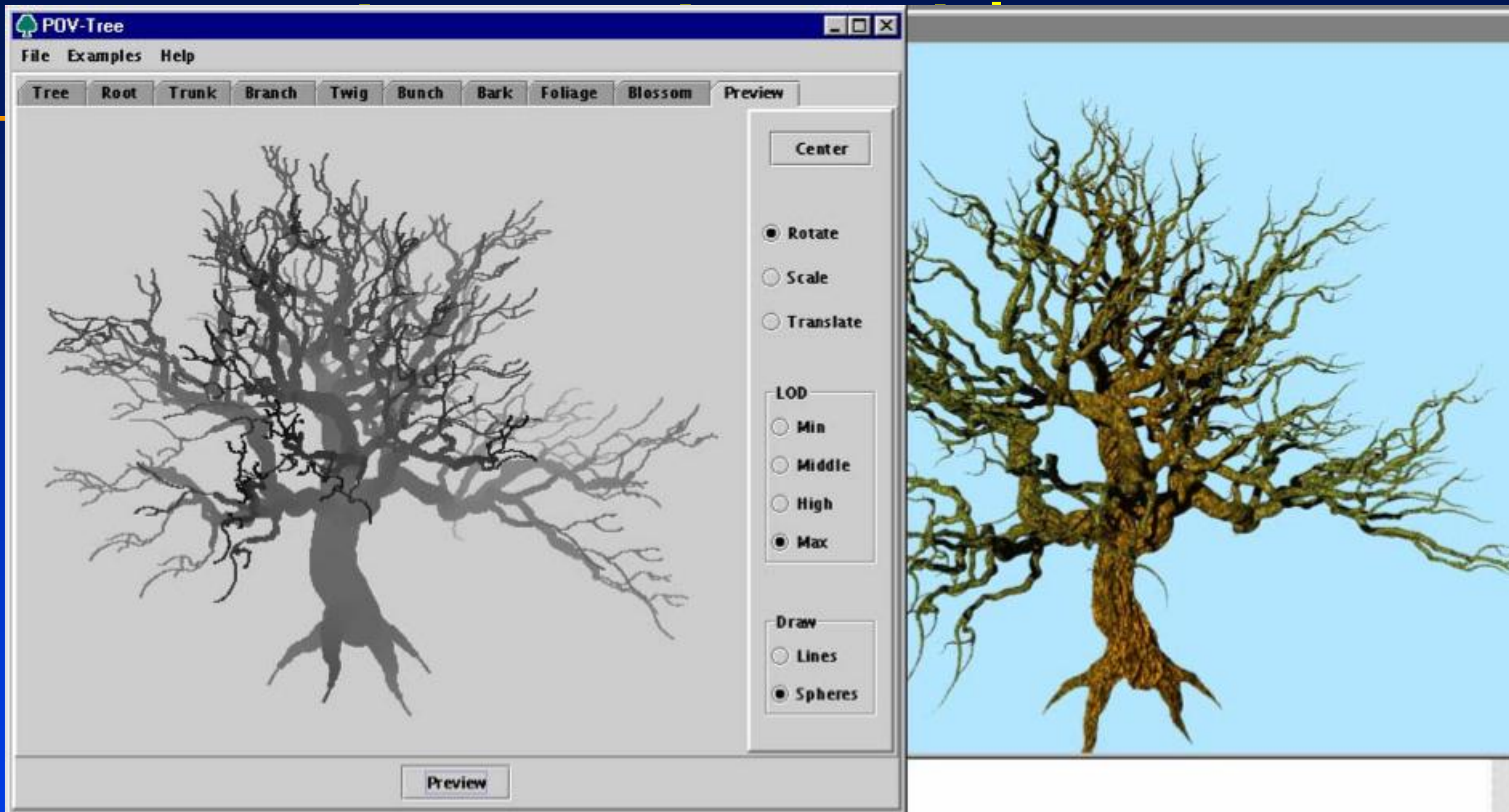
Algorithmic Botany

- http://algorithmicbotany.org/papers/
- **Free 200pg ebook**
- **Covers many variants of L-Systems, formal derivations, and exhaustive coverage of different plant types**



PovTree


Default
Birch
Poplar
Eucalyptus
Cherry
Spruce
Olive
Palm
Pine
Maple
Willow
Linden



- http://propro.ru/go/Wshop/povtree/povtree.html
- http://arbaro.sourceforge.net/

- **Fast procedural foliage is important for real-time applications**
- http://www.speedtree.com/

L-Systems: Further Readings

- Algorithmic Botany
  - Covers many variants of L-Systems, formal derivations, and exhaustive coverage of different plant types.
  - http://algorithmicbotany.org/papers
- PovTree
  - http://propro.ru/go/Wshop/povtree/povtree.html
  - http://arbaro.sourceforge.net/

Procedural Modeling

- Sweeps
- Fractals
- Grammars

Grammars

- Generate description of geometric model by applying production rules

$$S \rightarrow AB$$
$$A \rightarrow Ba \mid a$$
$$B \rightarrow Ab \mid b$$

AB
BaB
BaAb
AbaAb
.
.
.

Grammars

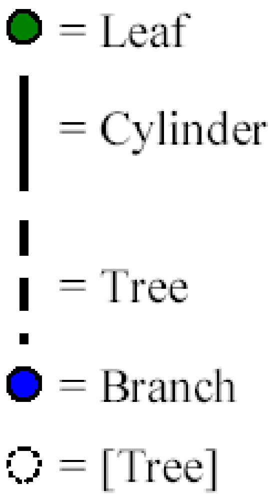
- Useful for creating plants

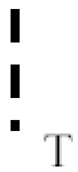
T

Start → Tree

Tree → Branch Tree | leaf

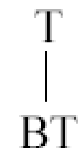
Branch → cylinder | [Tree]



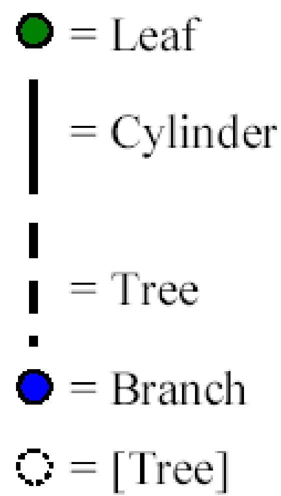


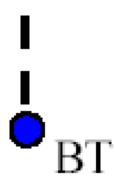
Grammars

- Useful for creating plants



Start → Tree
Tree → Branch Tree | leaf
Branch → cylinder | [Tree]

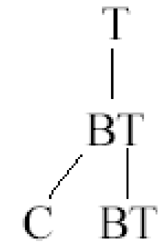


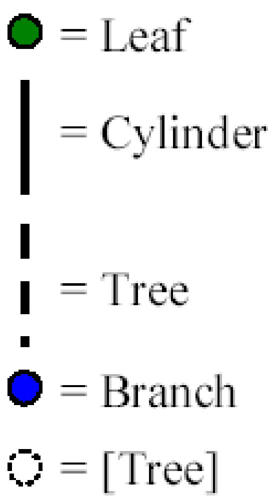


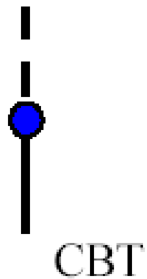
Grammars

- Useful for creating plants

Start → Tree
Tree → Branch Tree | leaf
Branch → cylinder | [Tree]

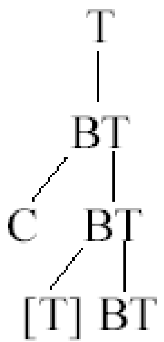




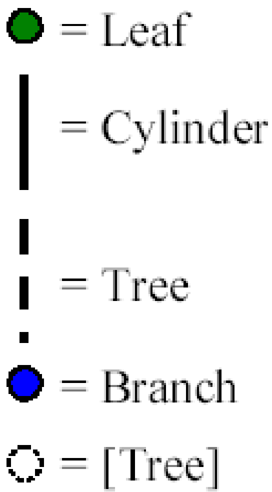


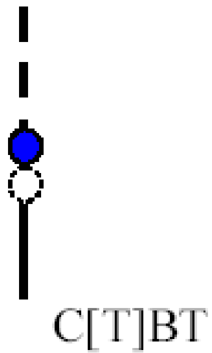
Grammars

- Useful for creating plants



Start $\longrightarrow$ Tree
Tree $\longrightarrow$ Branch Tree | leaf
Branch $\longrightarrow$ cylinder | [Tree]





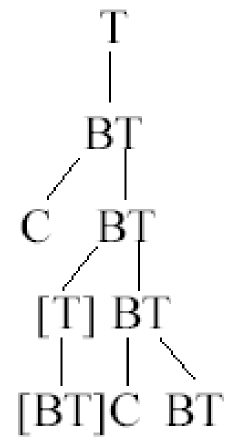
Grammars

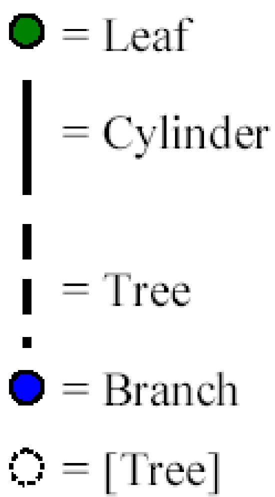
- Useful for creating plants

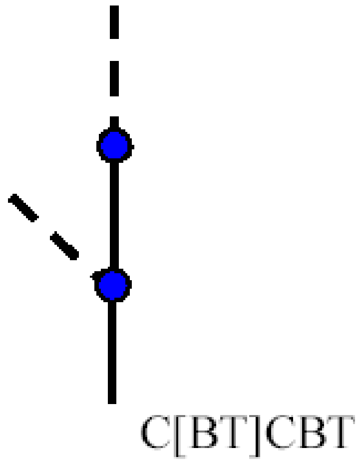
Start → Tree

Tree → Branch Tree | leaf

Branch → cylinder | [Tree]







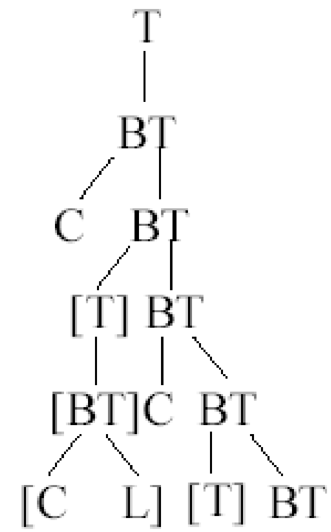
Grammars

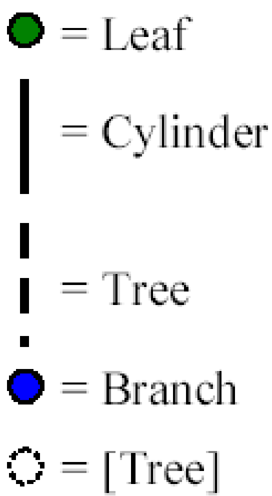
- Useful for creating plants

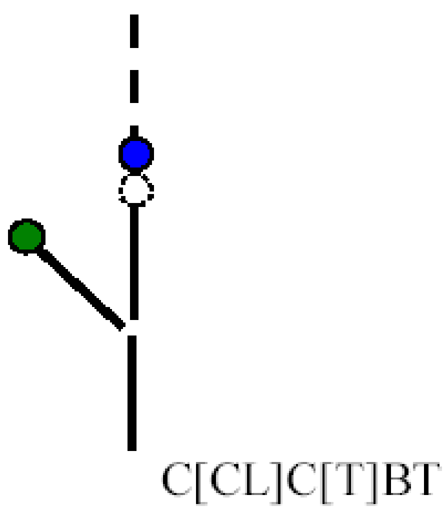
Start → Tree

Tree → Branch Tree | leaf

Branch → cylinder | [Tree]







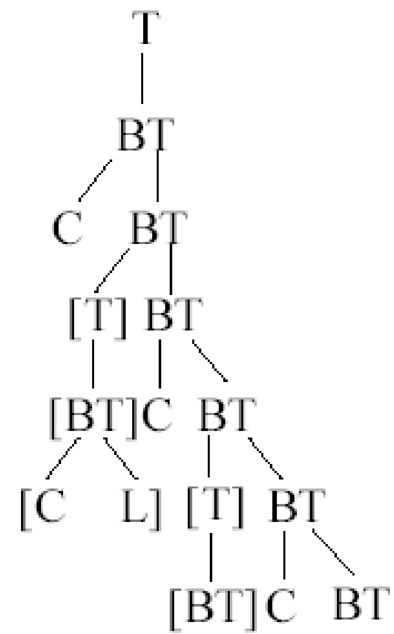
Grammars

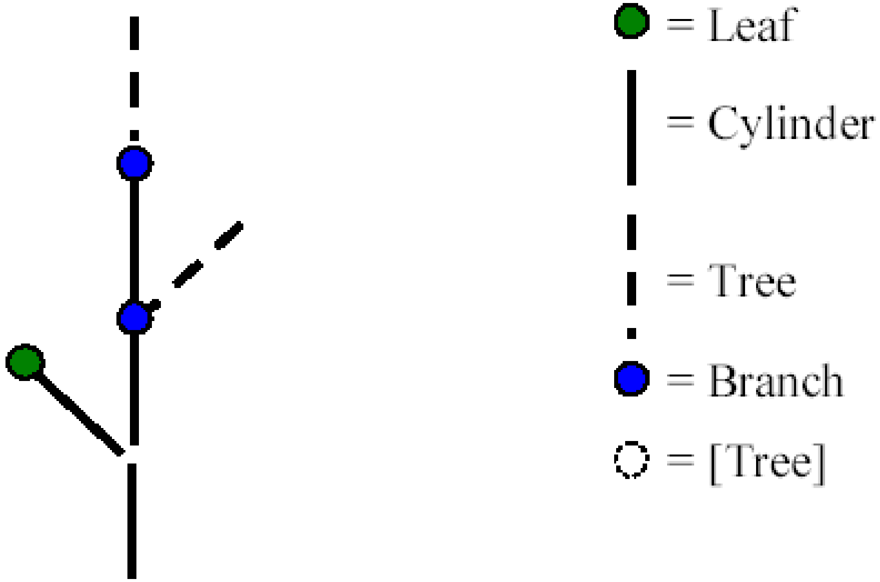
- Useful for creating plants

Start $\longrightarrow$ Tree

Tree $\longrightarrow$ Branch Tree | leaf

Branch $\longrightarrow$ cylinder | [Tree]



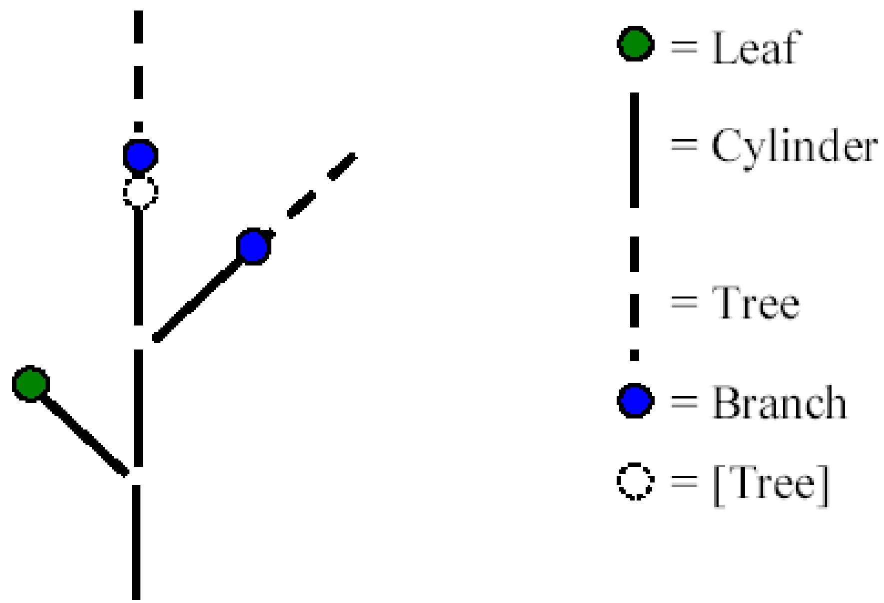


C[CL]C[BT]CBT

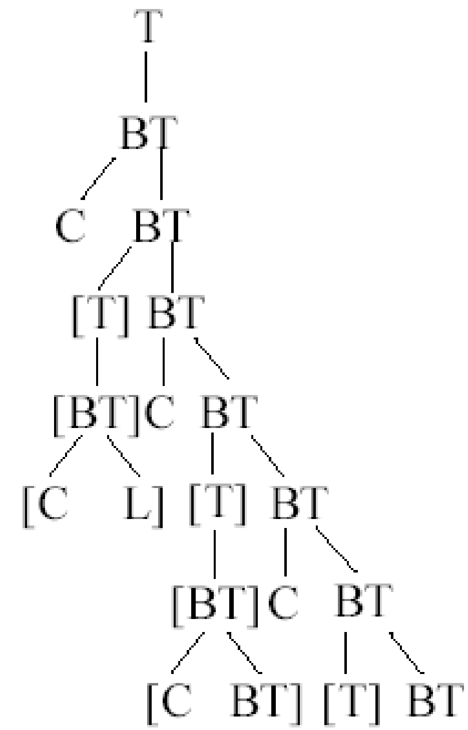
Grammars

- Useful for creating plants

Start → Tree
Tree → Branch Tree | leaf
Branch → cylinder | [Tree]



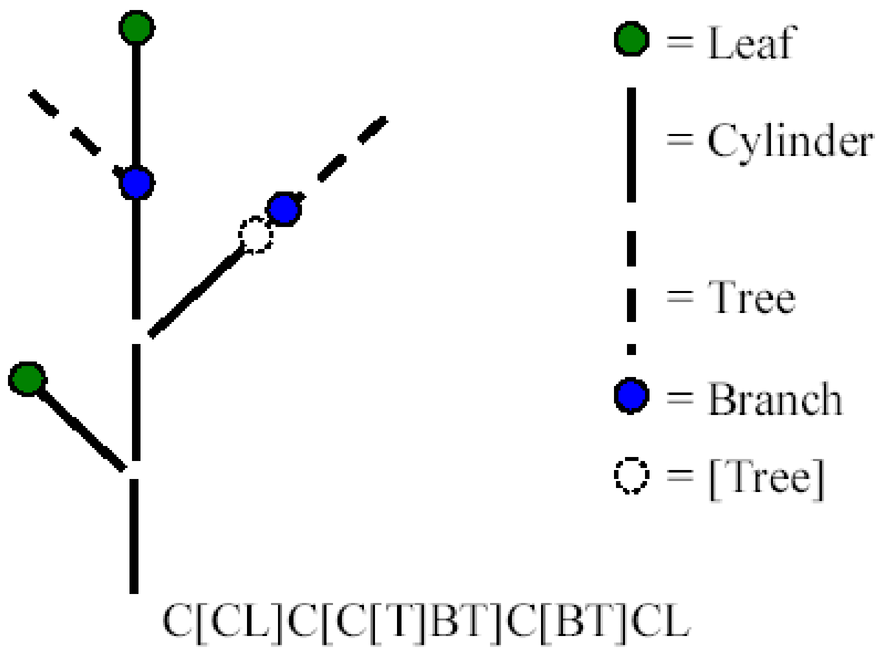
C[CL]C[CBT]C[T]BT

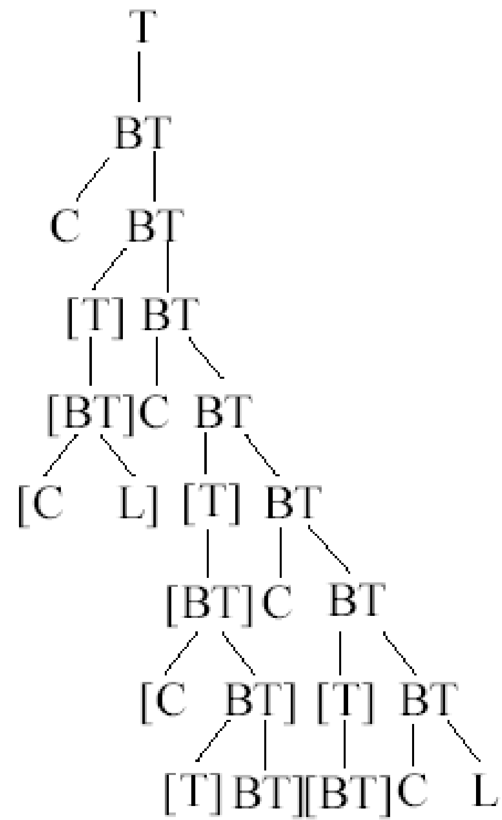


Grammars

- Useful for creating plants

Start → Tree
Tree → Branch Tree | leaf
Branch → cylinder | [Tree]

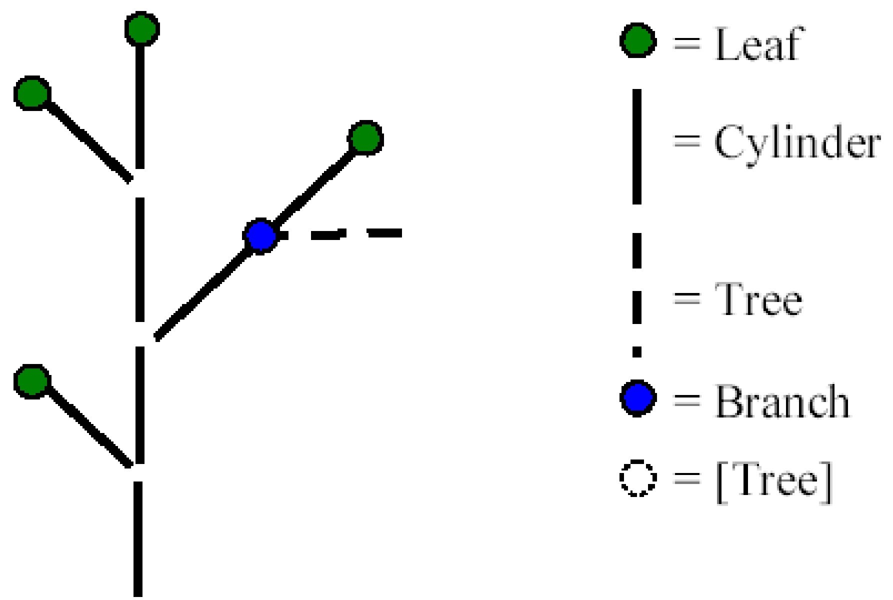




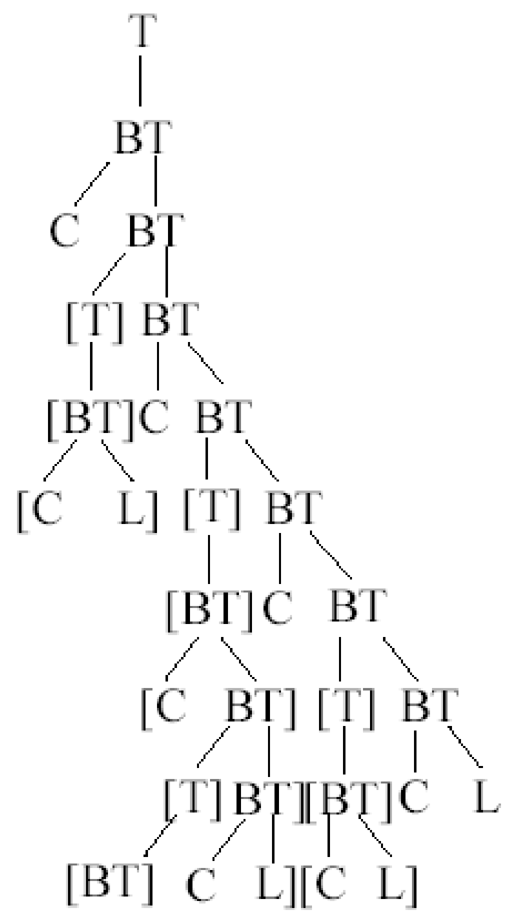
Grammars

- Useful for creating plants

Start → Tree
Tree → Branch Tree | leaf
Branch → cylinder | [Tree]



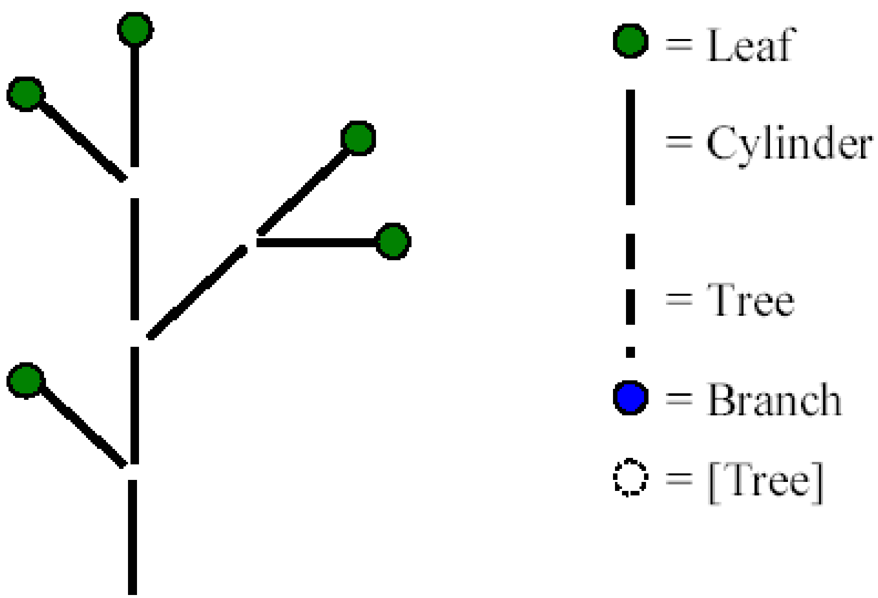
C[CL]C[C[BT]CL]C[CL]CL



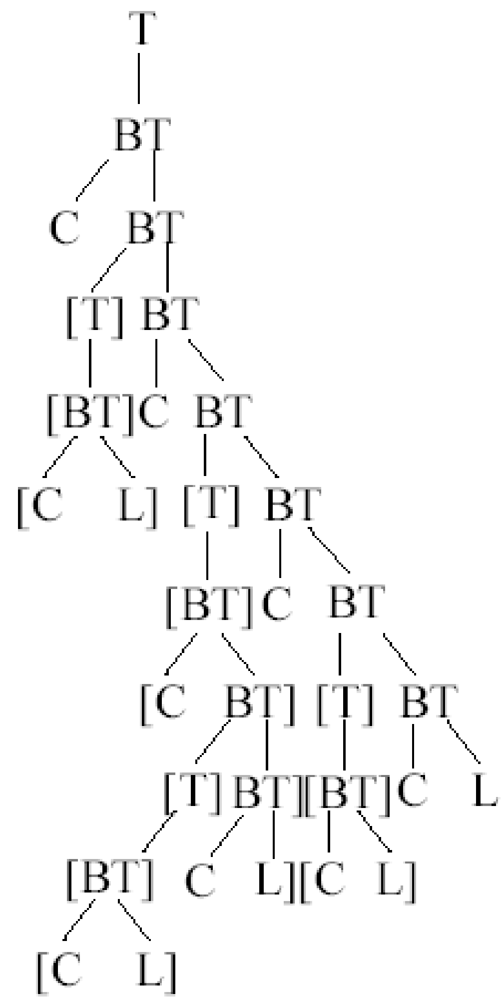
Grammars

- Useful for creating plants

Start → Tree
Tree → Branch Tree | leaf
Branch → cylinder | [Tree]

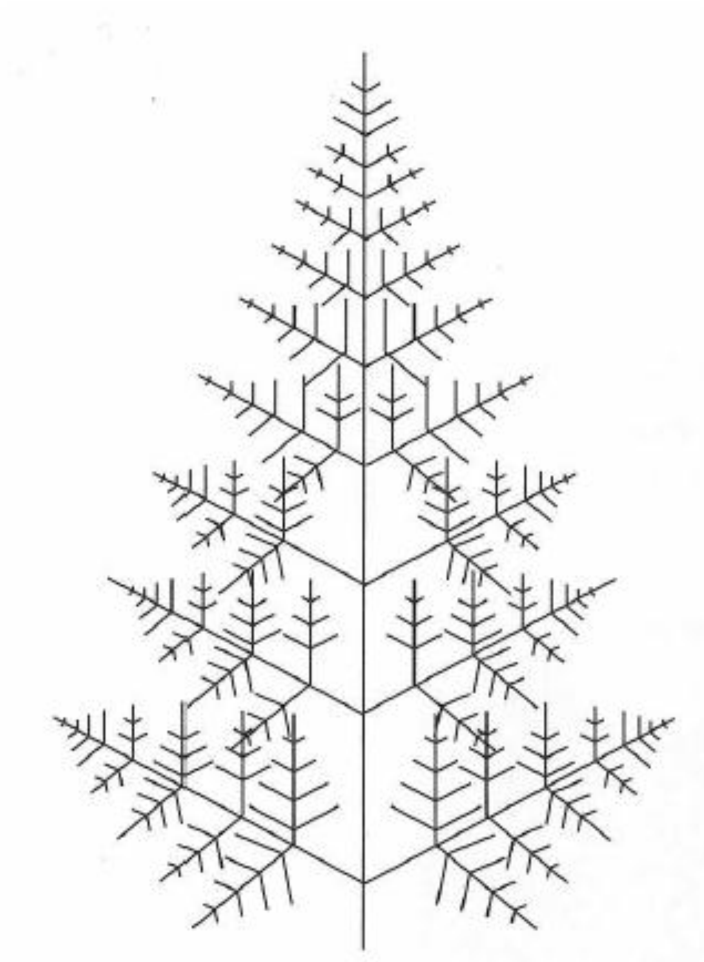


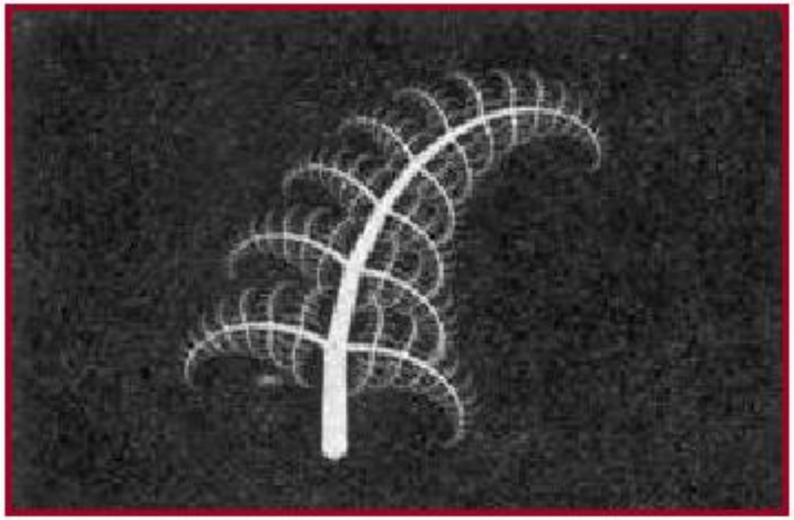
C[CL]C[C[CL]CL]C[CL]CL



Grammars

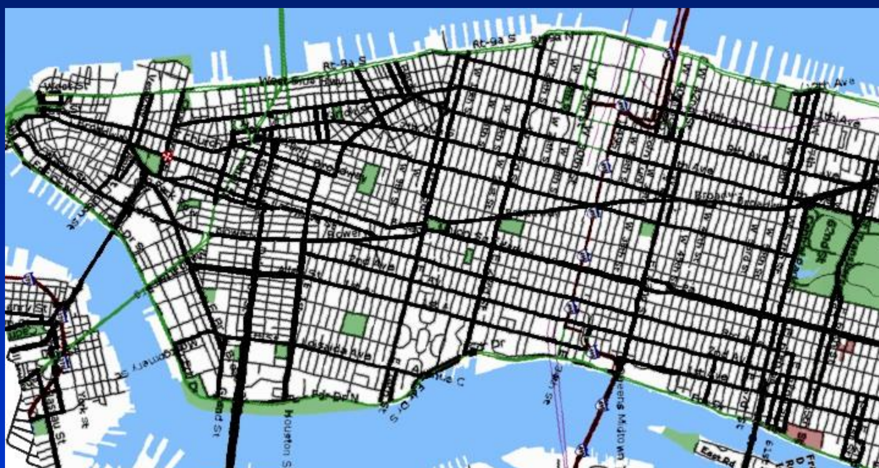
- Useful for creating plants



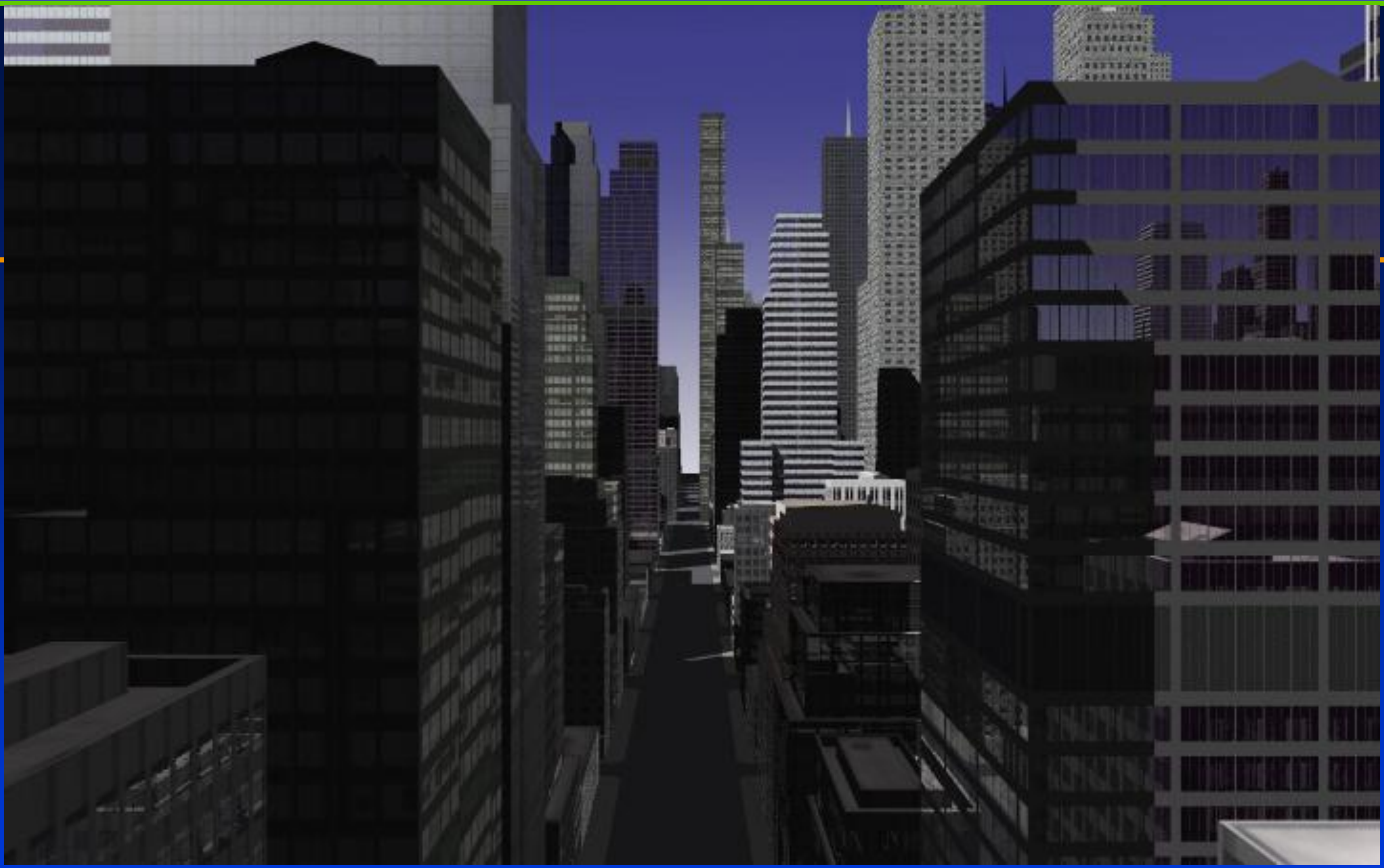


H&B Figure 10.77

L-Systems for Cities [Parish01]



- Start with a single street
- Branch & extend w/ parametric L-System
- Parameters of the string are tweaked by goals/constraints
- Goals control street direction, spacing
- Contraints allow for parks, bridges, road loops
- Once we have streets, we can form buildings with another L-System
- Building shapes are represented as CSG operations on simple shapes



- Once we have streets, we can form buildings with another L-System
- Building shapes are represented as CSG operations on simple shapes

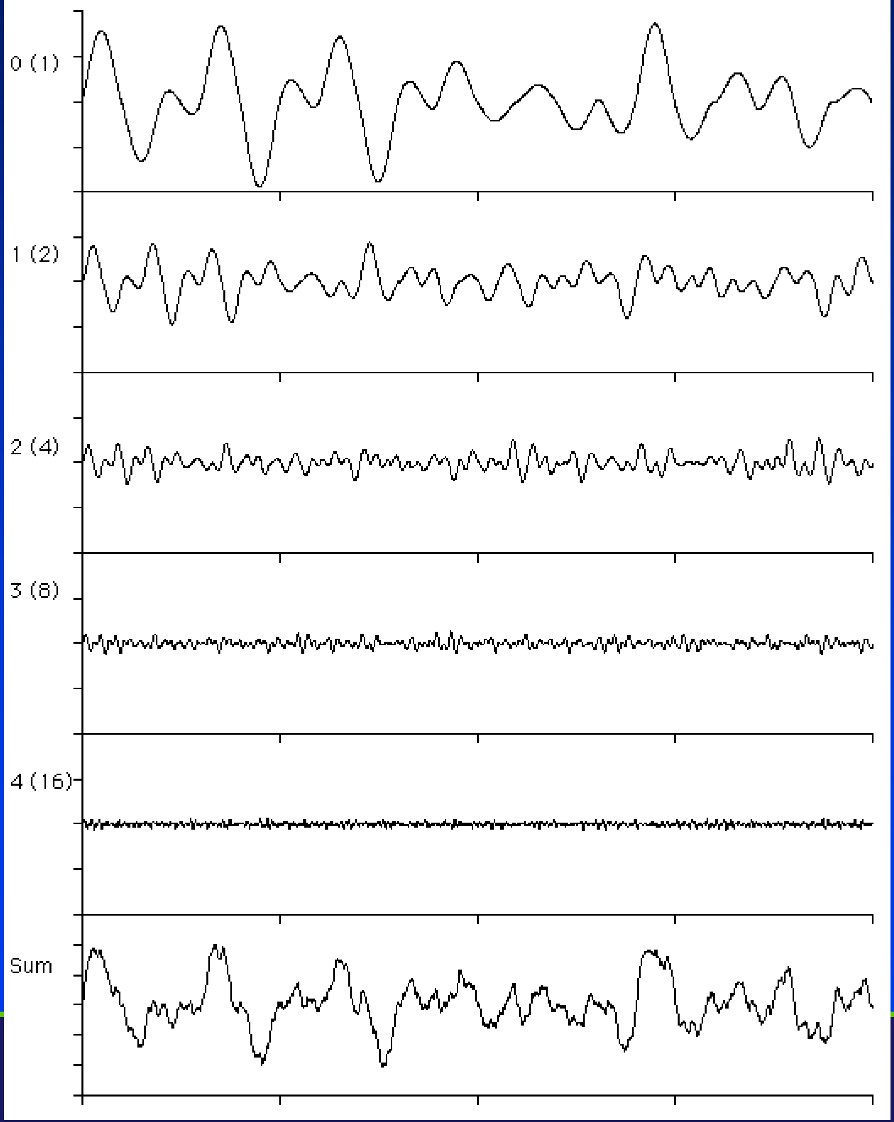
Procedural Terrain: Perlin Noise

- Noise Functions
  - Seeded pseudo-random number generator
  - Over R^n
  - Approximation to Gaussian filtered noise
  - Implemented as a pseudo-random spline
  - The trick is to make it fast

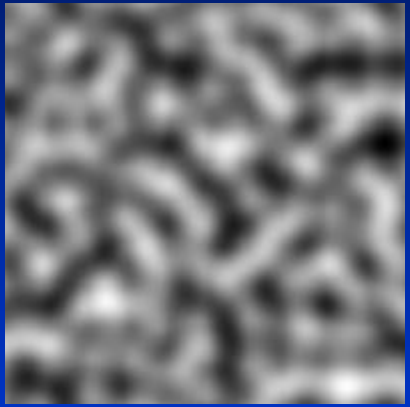
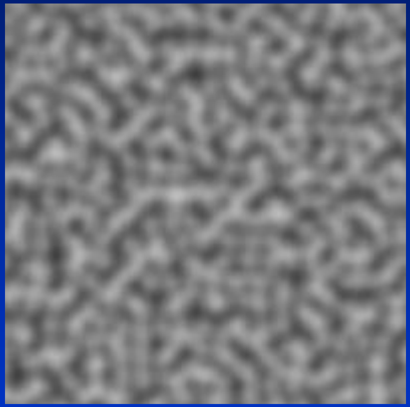
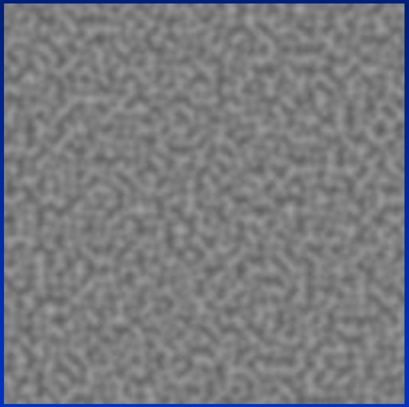
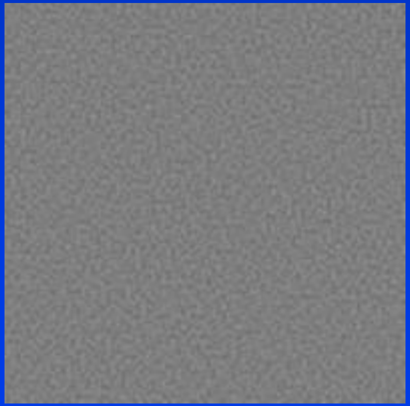
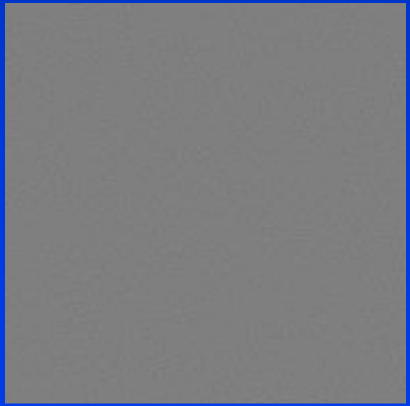
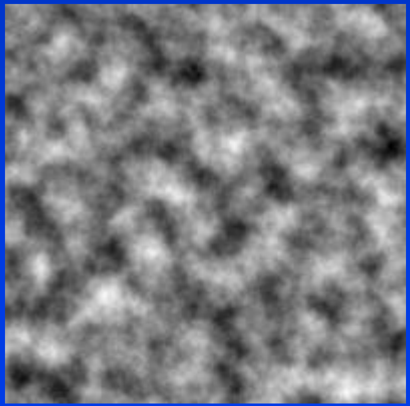
Noise Functions: Algorithm

- Given an input point
- For each of its neighboring grid points:
  - Pick a "pseudo-random" gradient vector
    - Pre-compute table of permutations P[n]
    - Pre-compute table of gradients G[n]
    - G = G[i + P[j + P[k]]]
  - Compute linear function (dot product)
- Take weighted sum, using ease curves
- http://www.noisemachine.com/talk1/java/noisegrid.html

Perlin Noises in 1-D

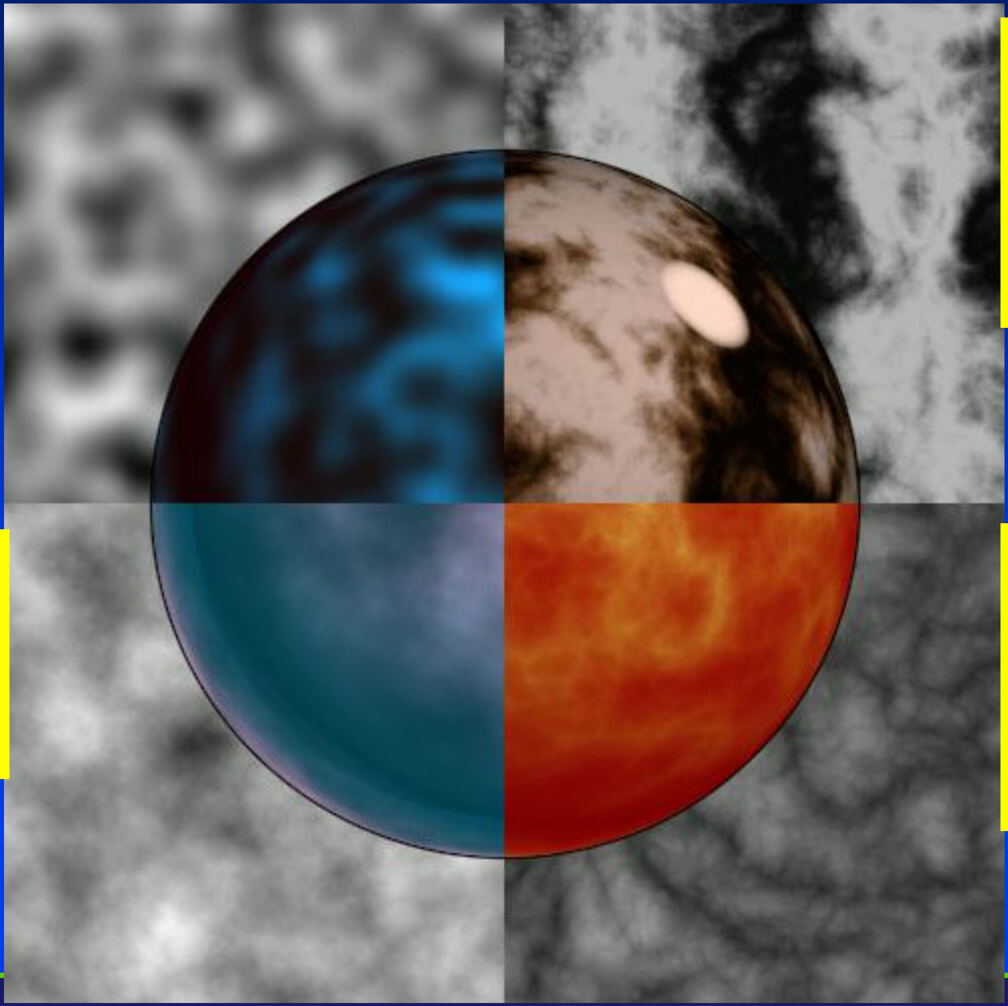


Perlin Noises in 2-D

Weighted Sums

noise:
-Worn metal
-Water wave
(gradient)



Sin(x + Sum[1/f *|noise|]):
-Turbulent flows
-Fire
-Marble

Sum[1/f * noise]:
-Rock
-Mountains
-Clouds

Sum[1/f * |noise|]:
-Turbulent flows
-Fire
-Marble
-Clouds

Using Noise in 3-D to Animate 2-D Flows

- Treating time as another spatial dimension
- Examples
  - Corona [K. Perlin]
    - http://www.noisemachine.com/talk1/imgs/flame500.html
  - Clouds [K. Perlin]
    - http://www.noisemachine.com/talk1/imgs/clouds500.html

“Implicit” vs. “Explicit” Procedural Models

- **Explicit approach:**
  - Directly generate the points that make up an object
  - Good for Z-buffer/OpenGL style rendering
- **Implicit approach:**
  - Answer questions about particular points
  - Isocurve (2D) or Isosurface (3D)
  - Good for ray-tracing/ray-casting

Hypertexture

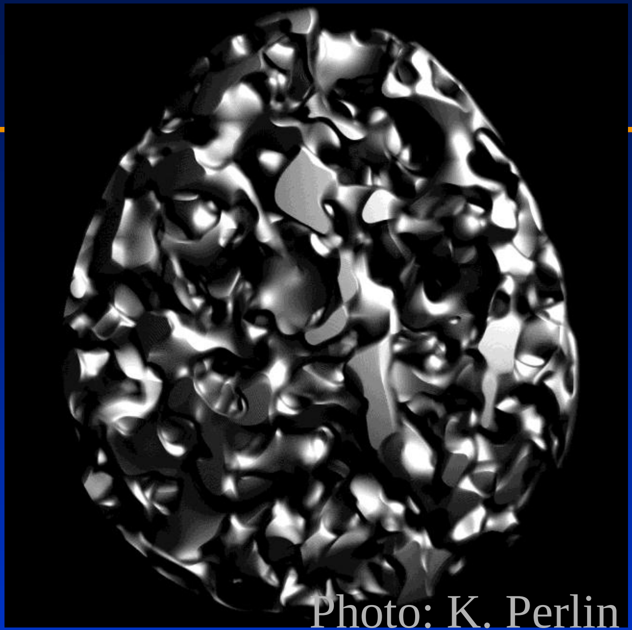
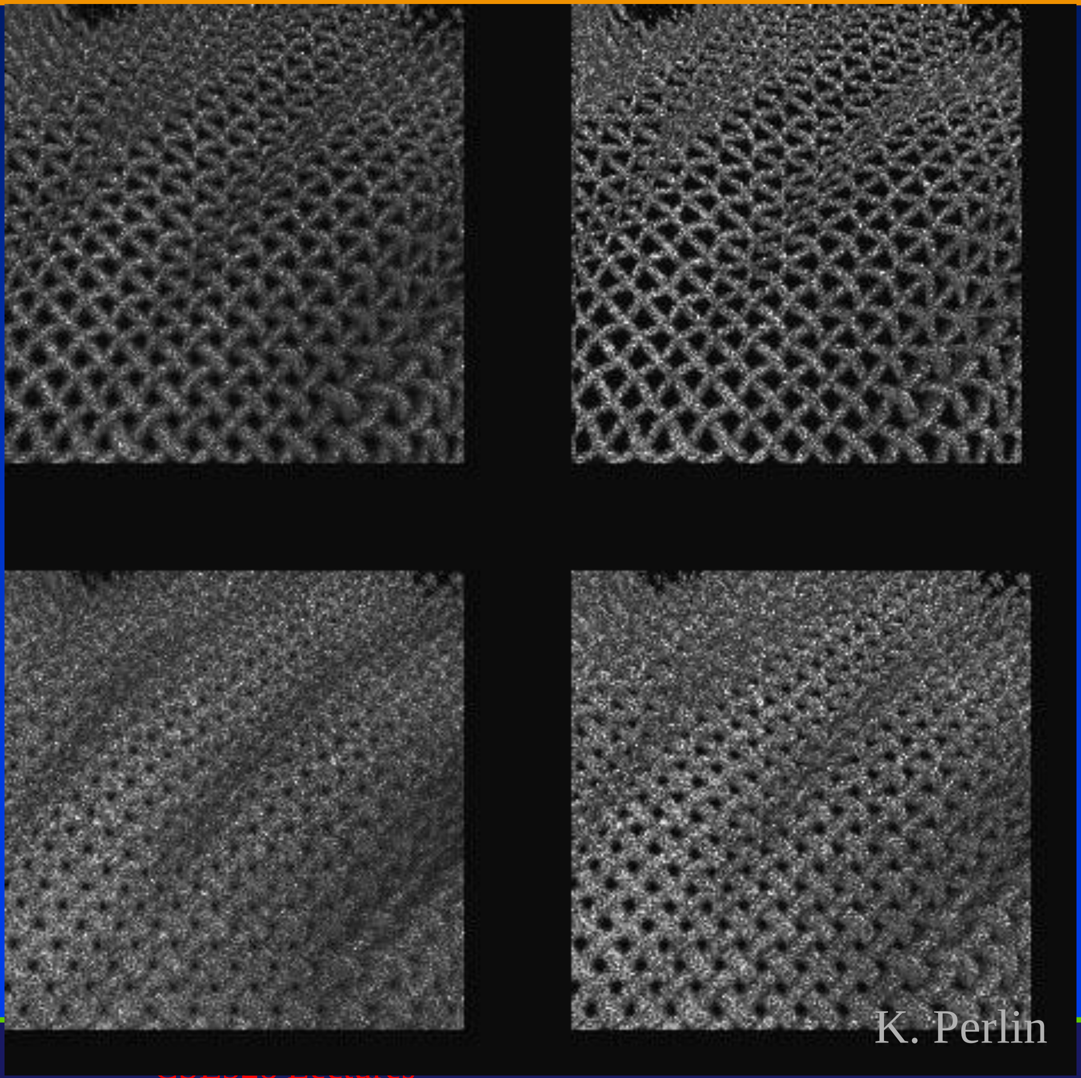


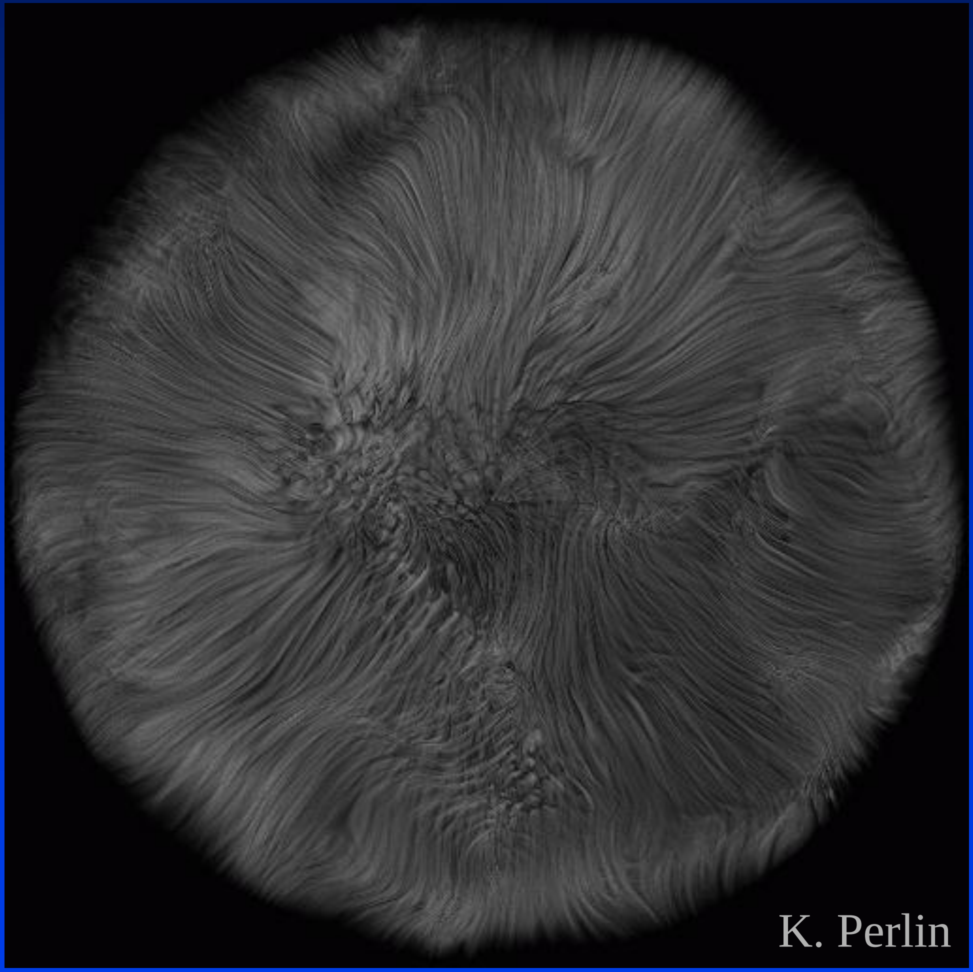
Photo: K. Perlin

- **Implicit procedural model**
- **Treat the isosurface of a function as the boundary of an object**
- **Above: fractal egg**

Hypertexture Example

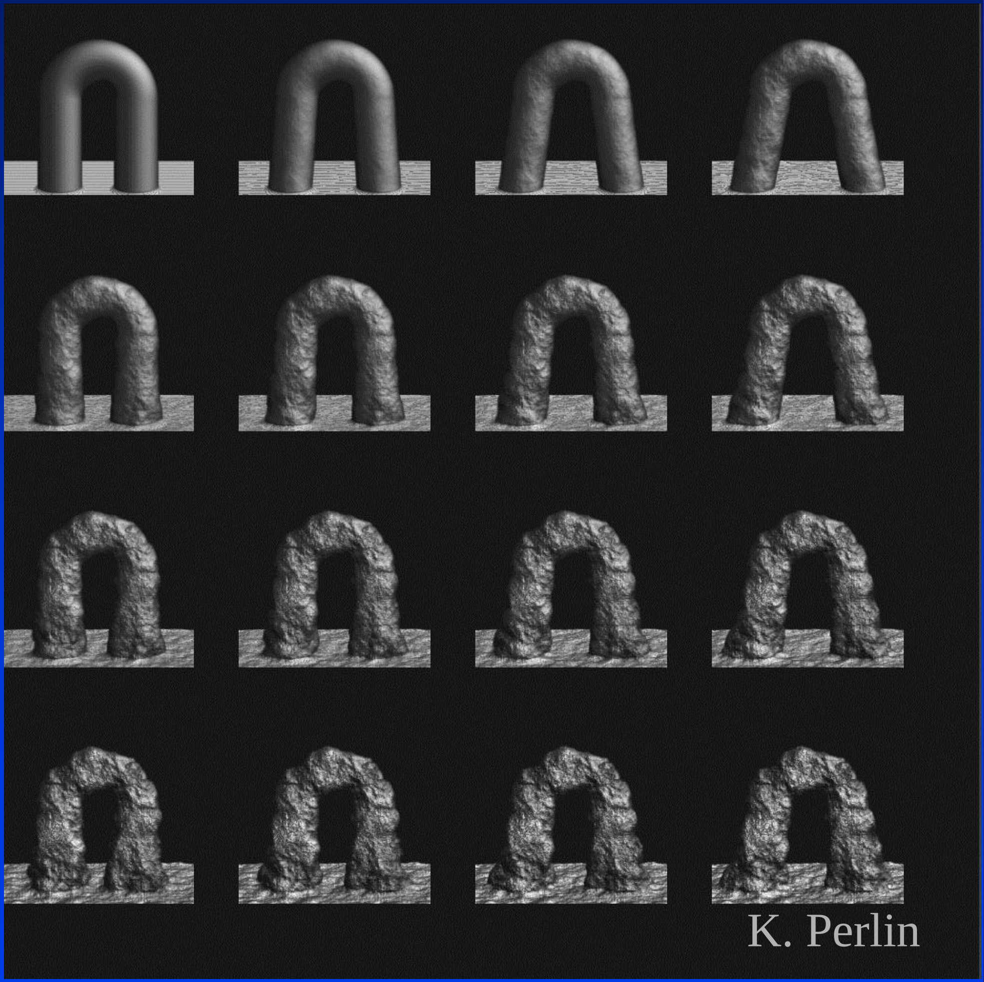


Hypertexture Example



Architexture

- **Sweep the path of a line drawing with a sphere**
- **Apply hypertexture to resulting shape**

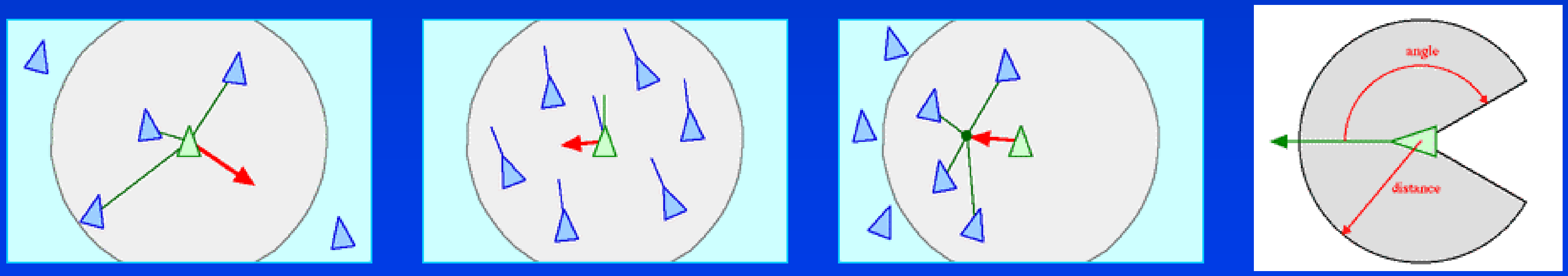


Procedural Animation

- **Particle Systems**
- **Ragdoll Physics**
- **Fluid simulation**
- **Flocking/Crowd Simulations**

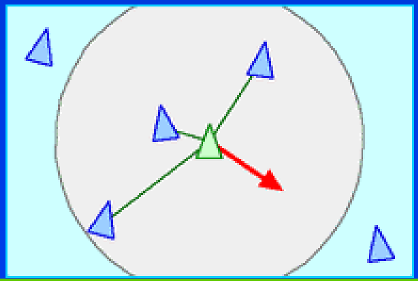
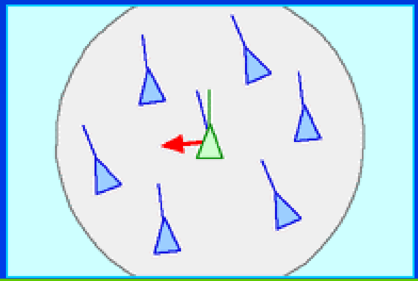
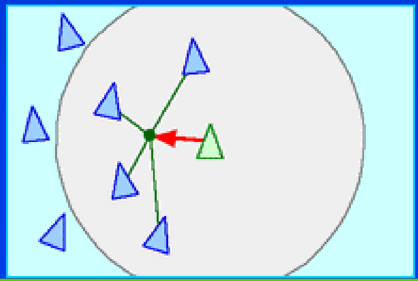
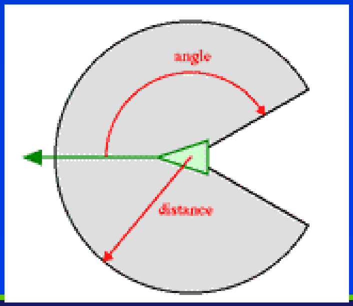
Procedural Flocking (Boids)

- Simulate the movement of a flock of birds in 3-space
- Separation: move to avoid crowding local neighbors
- Alignment: steer towards average heading of neighbors
- Cohesion: steer towards average position of neighbors
- Limited Senses: only neighbors in forward-facing arc are observable



Procedural Flocking (Boids)

- Simulate the movement of a flock of birds in 3-space
- Separation: move to avoid crowding local neighbors
- Alignment: steer towards average heading of neighbors
- Cohesion: steer towards average position of neighbors
- Limited Senses: only neighbors in forward-facing arc are observable

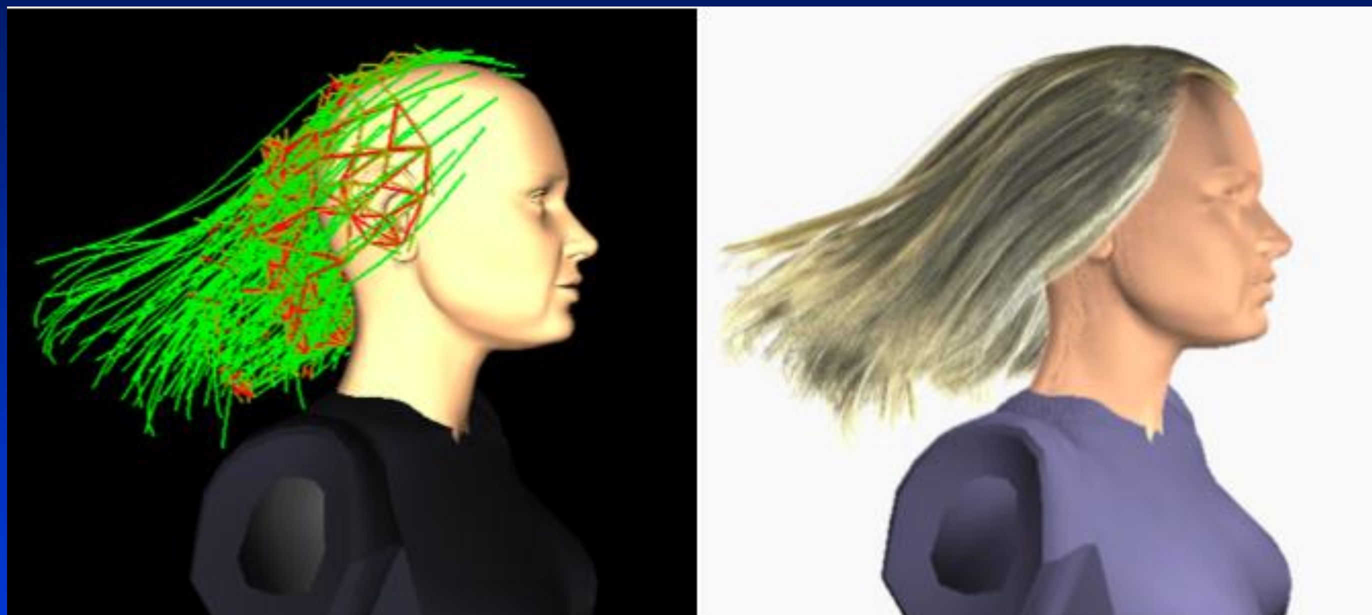
Boids Sample

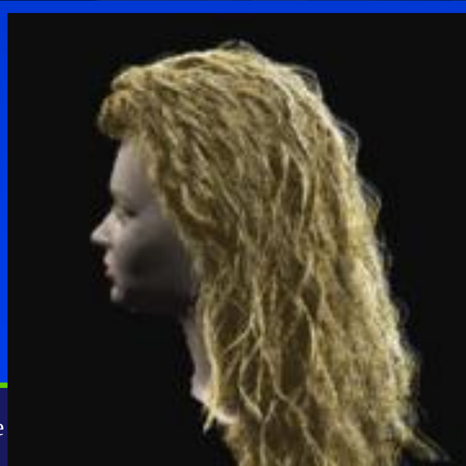
- http://www.red3d.com/cwr/boids/
- http://www.siggraph.org/education/materials/HyperGraph/animation/art_life/video/3cr.mov

Procedural Hair [Chang02]

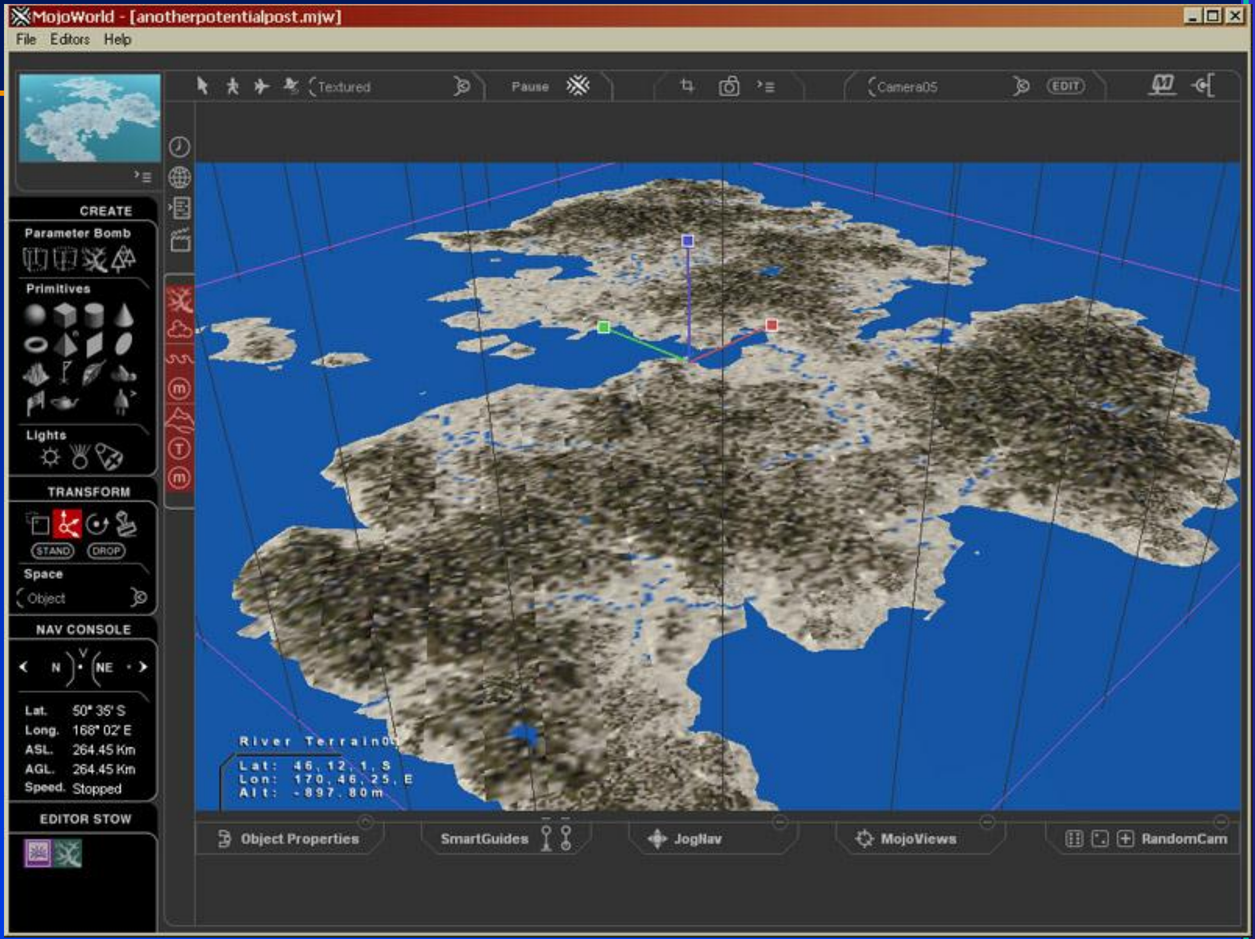
- **Generate a model with a few hundred guide hairs**
- **Each hair is a rigid chain w/ revolute joints**
- **Use breakable springs between nearby hairs to simulate hairstyles**
- **Create triangle strips between adjacent hairs to simulate collisions**
- **Interpolate between guide hairs to produce many other hairs**

Procedural Hair (Examples)





MojoWorld



- **Commercial application for creating photorealistic procedural planets**

http://www.pandromeda.com/

Procedural Planets

Procedural Planets

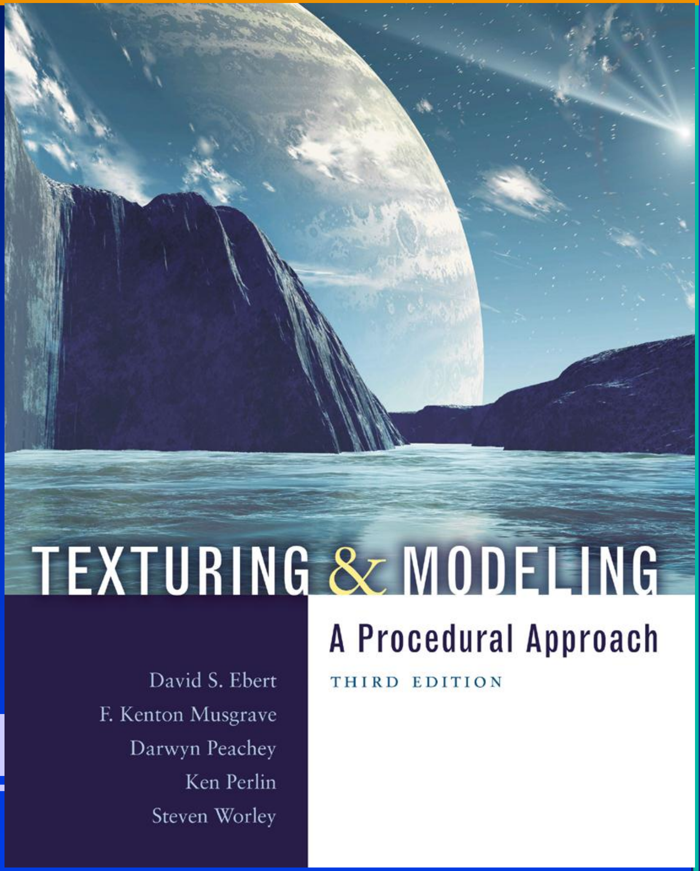
Procedural Planets

F.K. Musgrave

Texturing and Modeling: A Procedural Approach

- **D.S. Ebert et al**
- **3rd Ed, 2003**
- **Excellent reference**

- http://www.mkp.com/tm3
- http://www.texturingandmod



Boids Example

- **Open example**

Flow-Based Video Synthesis And Editing [Bhat04]

- **Allows animator to easily create loops and variants of flowing natural phenomena (water, smoke, etc.)**
- **Artist draws a set of flow lines on the original image**
- **Algorithm computes textures for a particle system that uses these flow lines**
- **Sequence of textures is transformed to prevent linear discontinuities**
- **Artist can then draw additional flow lines to create new variants**

Flow-Based Synthesis (Example)

Flow-Based Synthesis (Example)

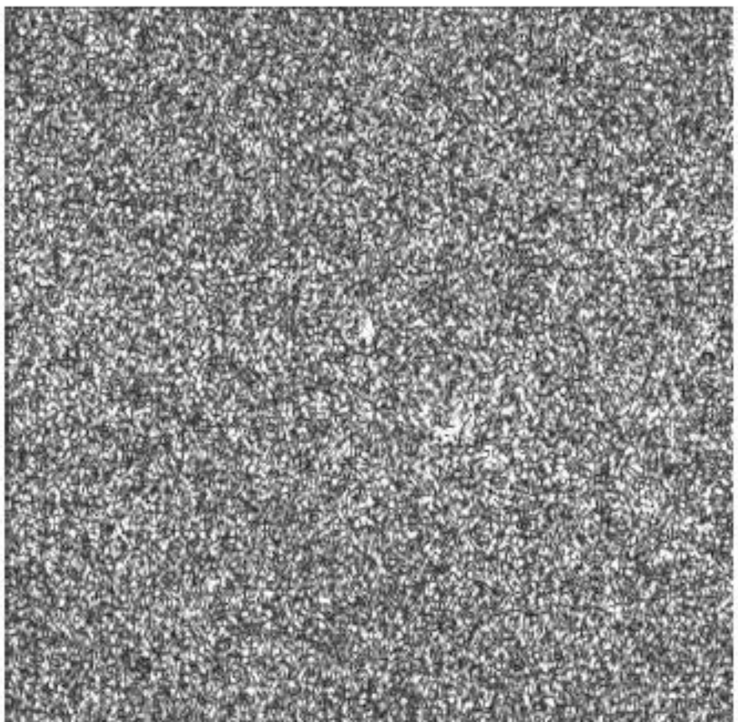
Procedural Modeling

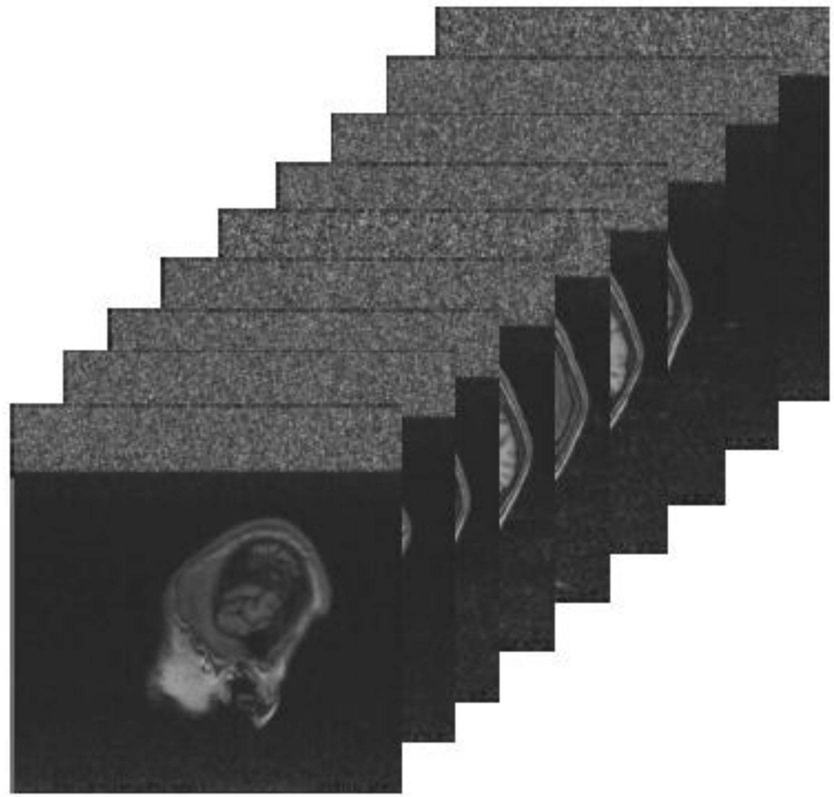
Model Construction

- Interactive modeling tools
  - CAD programs
  - Subdivision surface editors
- Scanning tools
  - CAT/MRI, Laser, robotic arm, etc.
- Computer vision
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Scanning tools

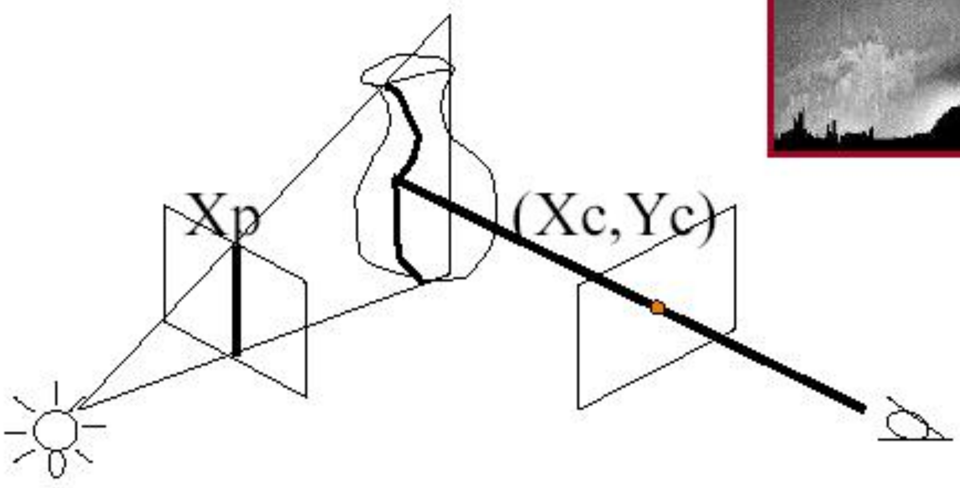
- Acquire geometry of objects with active sensors
  - CAT/MRI
  - Laser range scanner
  - Robotic arm





Scanning tools

- Acquire geometry of objects with active sensors
  - CAT/MRI
  - Laser range scanner
  - Robotic arm
  - etc.

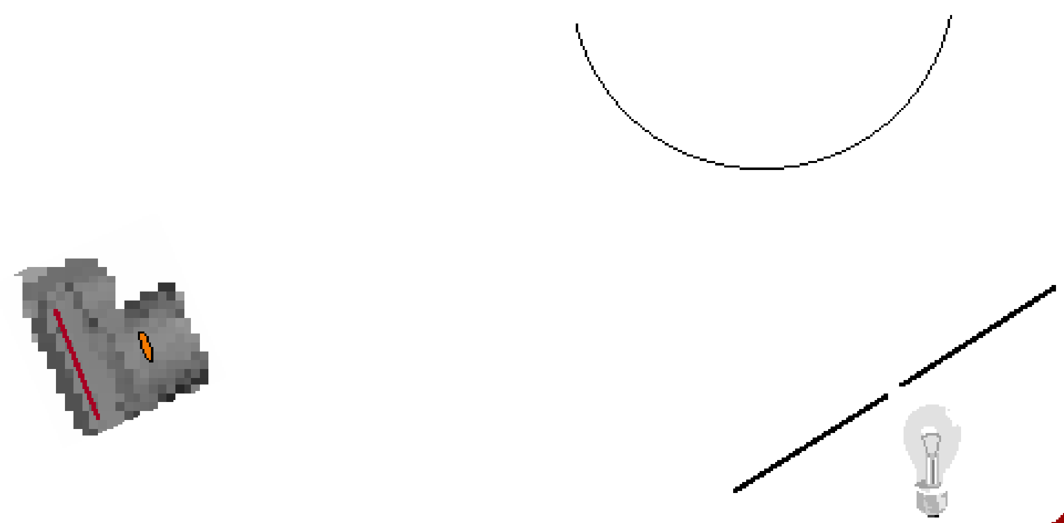


Scanning tools

Triangulation (in 2D):

To figure out the position of a point we need:

1. The position of the camera.
2. The position of the light source



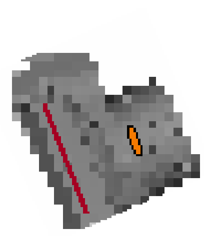
Scanning tools

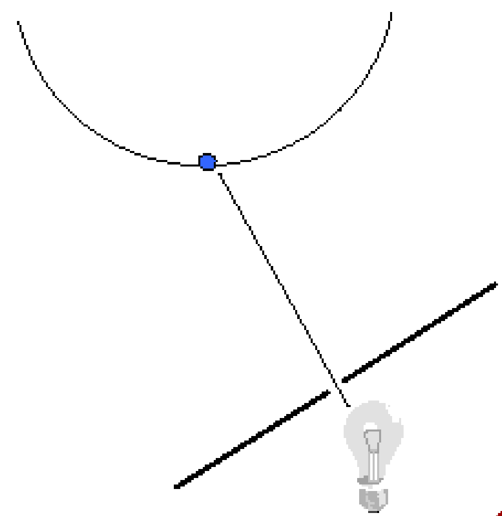
Triangulation (in 2D):

To figure out the position of a point we need:

1. The position of the camera
2. The position of the light source

Project the light onto the surface…





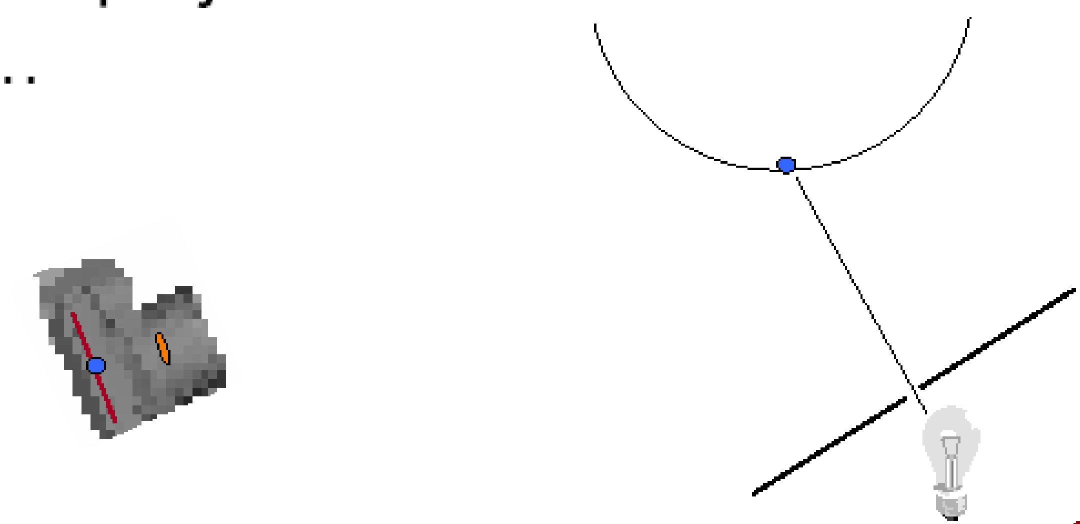
Scanning tools

Triangulation (in 2D):

To figure out the position of a point we need:

1. The position of the camera
2. The position of the light source

Project the light onto the surface…
Find where the lit point projects
onto the camera…



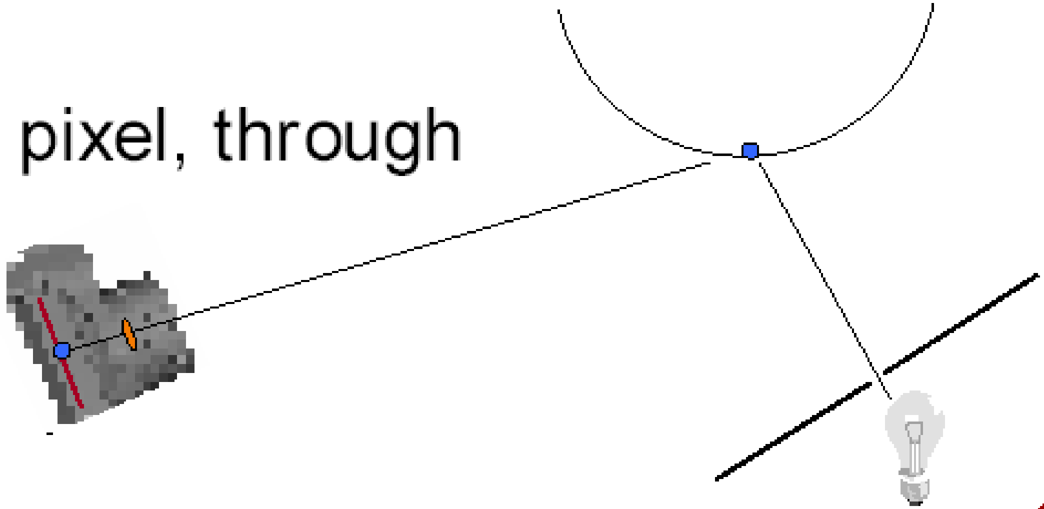
Scanning tools

Triangulation (in 2D):

To figure out the position of a point we need:

1. The position of the camera
2. The position of the light source

Project the light onto the surface…
Find where the lit point projects onto the camera…
Cast a ray from the lit pixel, through the camera…



Scanning tools

Triangulation (in 2D):

To figure out the position of a point we need:

1. The position of the camera
2. The position of the light source

Project the light onto the surface…

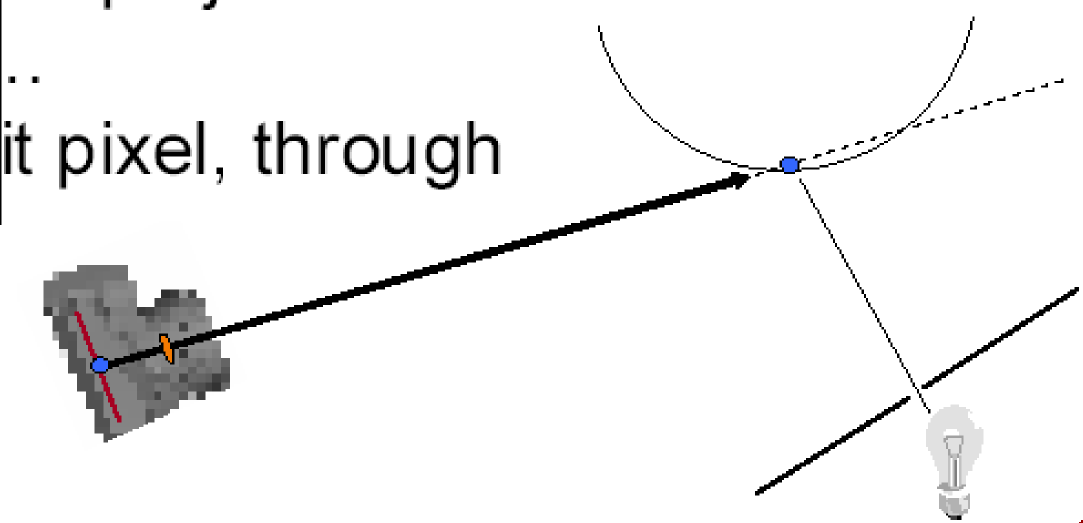
For the lit point to project onto the appropriate pixel, it has to lie somewhere on the ray.

int projects

…

it pixel, through

the camera…



Scanning tools

Triangulation (in 2D):

To figure out the position of a point we need:

1. The position of the camera
2. The position of the light source

Project the light onto the surface…

int projects

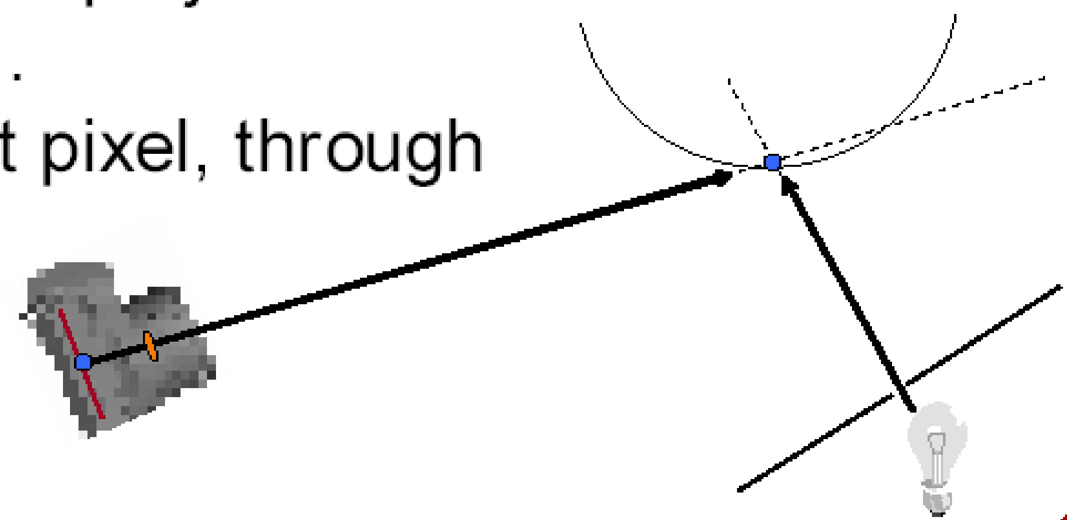
…

it pixel, through

the camera

For the lit point to project onto the appropriate pixel, it has to lie somewhere on the ray.

The lit point is also constrained to lie on the ray from the light source.



Scanning tools

Triangulation (in 2D):

To figure out the position of a point we need:

1. The position of the camera
2. The position of the light source

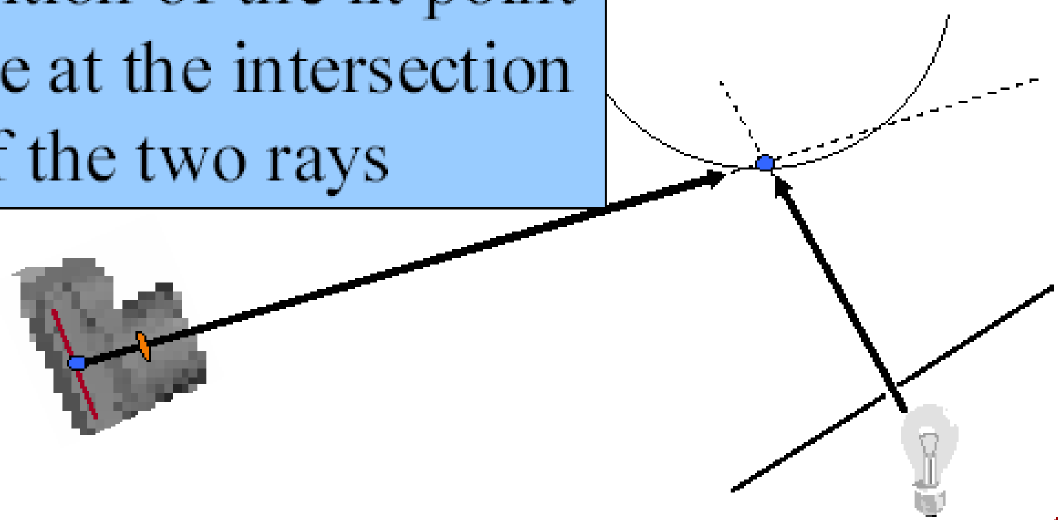
Project the light onto the surface…

For the lit point to
onto the appropria
it has to lie somew
the ray.

The position of the lit point has to be at the intersection of the two rays

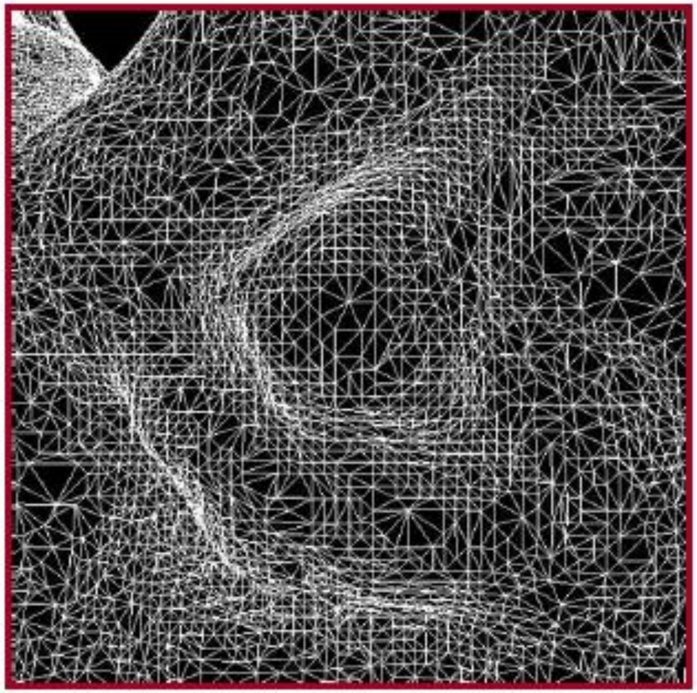
the camera

The lit point is also constrained to lie on the ray from the light source.



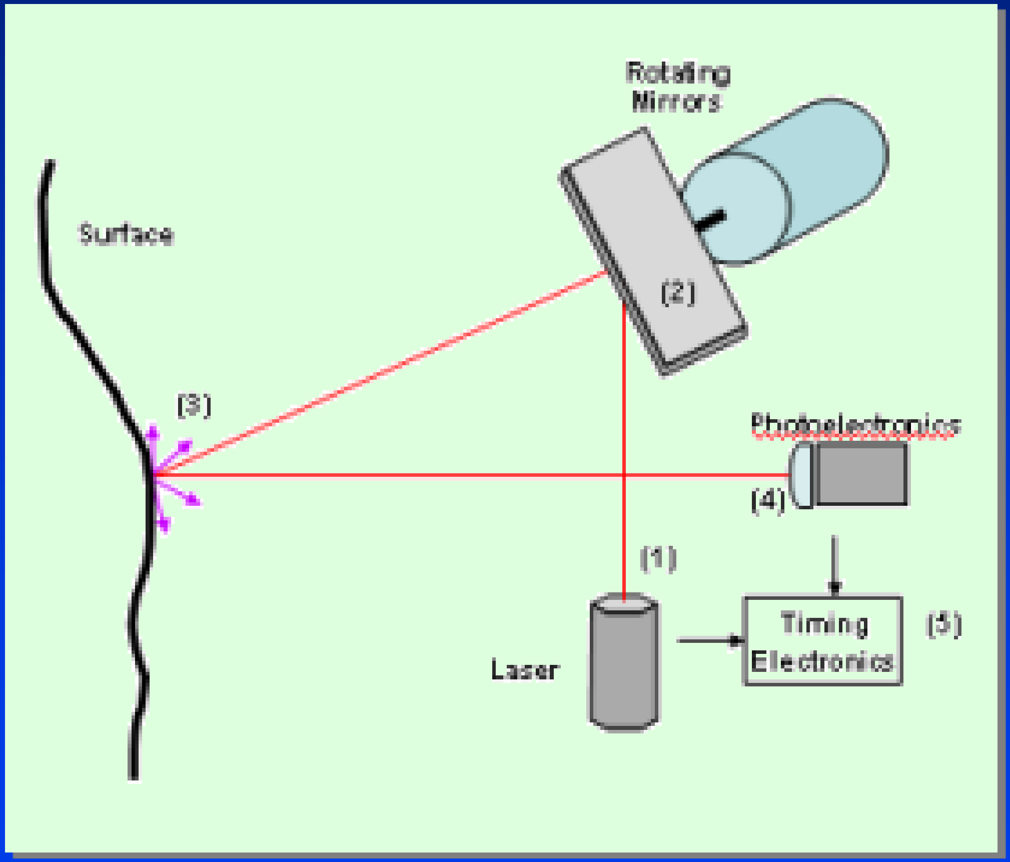
Laser Range Scanning

- Example: 70 scans



Stanford Graphics Laboratory

Other Range Scanning Tool



Scanning tools

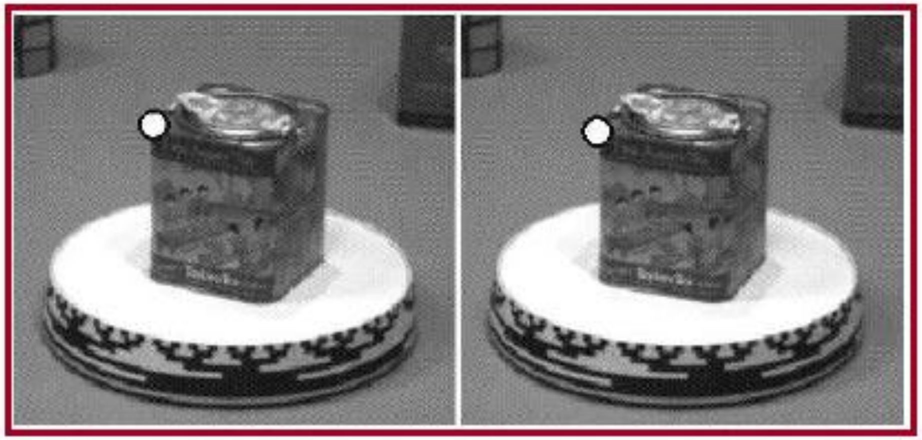
- Acquire geometry of objects with active sensors
  - CAT/MRI
  - Laser range scanner
  - Magnetic sensor
  - Robotic arm
  - etc.

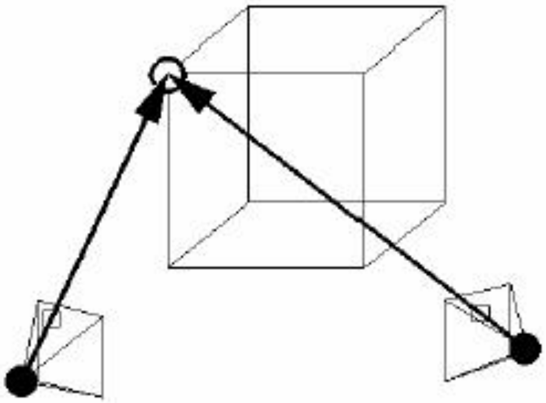
Model Construction

- Interactive modeling tools
  - CAD programs
  - Subdivision surface editors :)
- Scanning tools
  - Laser, magnetic, robotic arm, etc.
- Computer vision
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Computer Vision

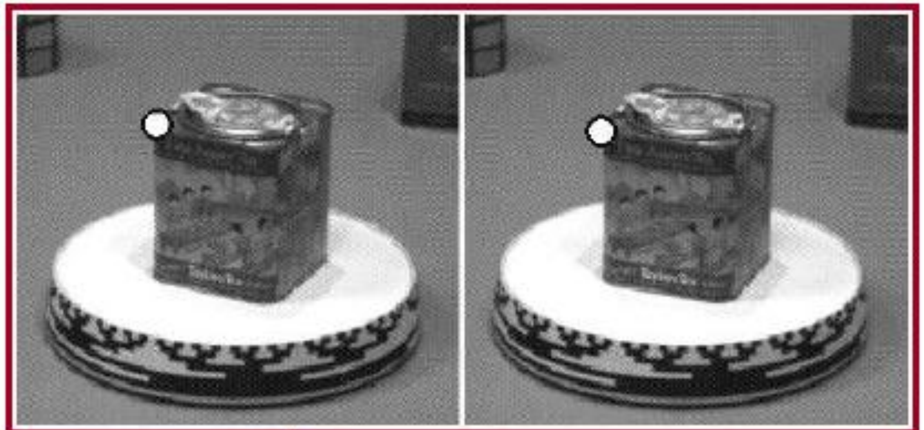
- Infer 3D geometry from images
  - Stereo
  - Motion
  - etc.



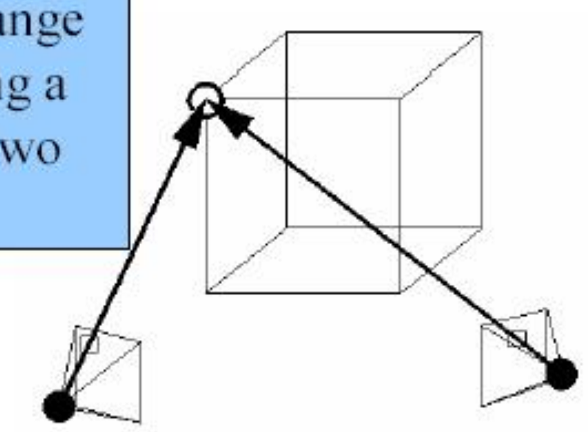


Computer Vision

- Infer 3D geometry from images
  - Stereo
  - Motion
  - etc.

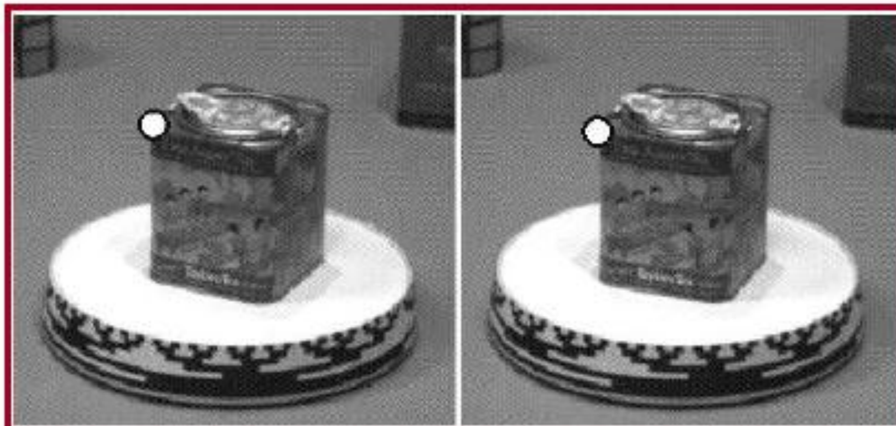


This is similar to the approach for laser range scanners, but instead of triangulating using a light and a camera, we triangulate using two cameras.



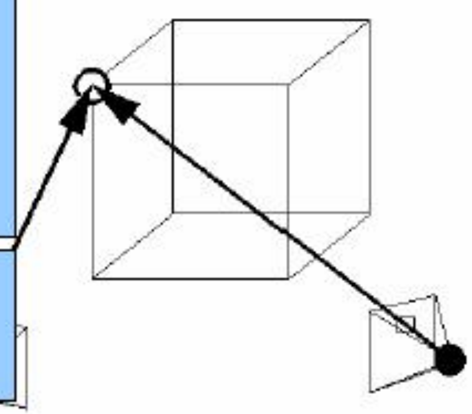
Computer Vision

- Infer 3D geometry from images
  - Stereo
  - Motion
  - etc.



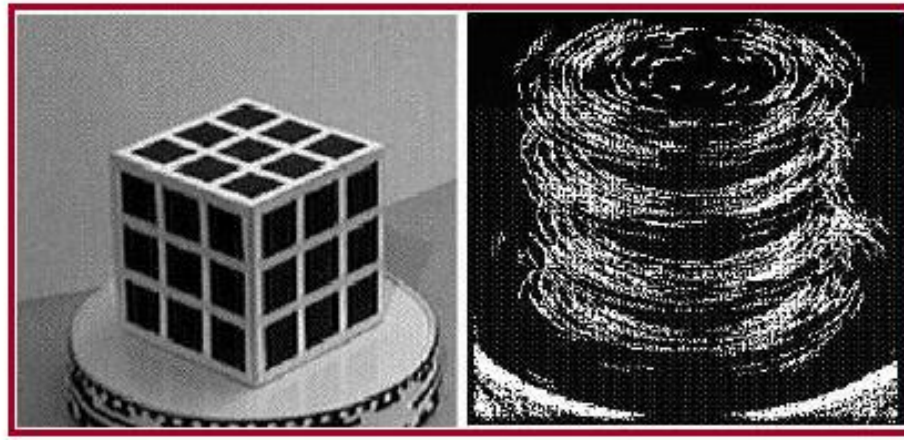
This is similar to the approach for laser range scanners, but instead of triangulating using a light and a camera, we triangulate using two cameras.

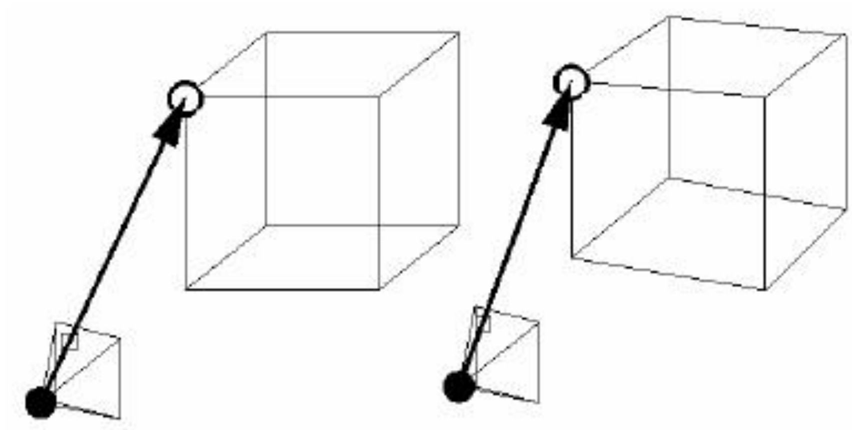
The challenge is to determine pixel pair correspondences across the two images.



Computer Vision

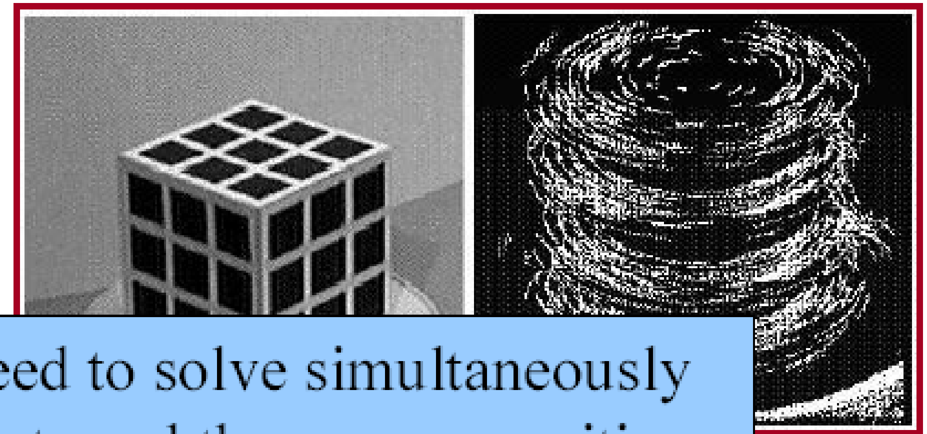
- Infer 3D geometry from images
  - Stereo
  - Motion
  - etc.



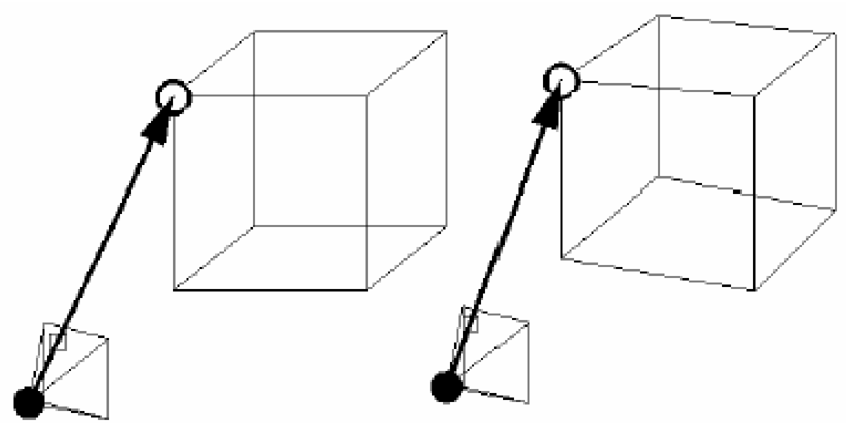


Computer Vision

- Infer 3D geometry from images
  - Stereo
  - Motion
  - etc.



In this case we need to solve simultaneously for the surface points and the camera position.



Model Construction

- Interactive modeling tools
  - CAD programs
  - Subdivision surface editors :)
- Scanning tools
  - Laser, magnetic, robotic arm, etc.
- Computer vision
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Summary

- Interactive modeling tools
  - CAD programs
  - Subdivision surface editors
- Scanning tools
  - CAT, MRI, Laser, magnetic, robotic arm, etc.
- Computer vision
  - Stereo, motion, etc.
- Procedural generation
  - Sweeps, fractals, grammars

Constructing 3D models is hard!

Jurassic Park